



MPC5604E Microcontroller Reference Manual

Devices Supported:
MPC5604E

MPC5604ERM
Rev. 5
07/2015

This page is intentionally left blank.

Chapter 1 Overview

1.1	Chipset overview	9
1.2	Target applications	10
1.3	Features	10
1.4	Block diagram	11
1.5	Application examples	13
1.5.1	CMOS vision sensor gateway	13
1.6	Audio source gateway	16
1.7	Critical performance parameters	18
1.8	Chip-level features	18
1.8.1	High performance e200z0 core CPU	19
1.8.2	Crossbar switch (XBAR)	20
1.8.3	System clocks and clock generation	20
1.8.4	Frequency Modulated Phase Lock Loop (FMPLL)	21
1.8.5	Main oscillator	21
1.8.6	Internal RC oscillator	21
1.8.7	Voltage regulator (VREG)	22
1.8.8	System Integration Unit (SIU-Lite)	22
1.8.9	Boot Assist Module (BAM)	22
1.8.10	Junction temperature sensor	23
1.8.11	JTAG controller (JTAGC)	23
1.8.12	Nexus Debug Interface (NDI)	24
1.8.13	DMA controller	24
1.8.14	DMA channel multiplexer (DMA_MUX)	25
1.8.15	Software Watchdog Timer (SWT)	25
1.8.16	System Timer Module (STM)	25
1.8.17	Periodic Interrupt Timers (PIT)	26
1.8.18	FlexCAN module	26
1.8.19	Deserial Serial Peripheral Interface (DSPI)	27
1.8.20	Serial communication interface module (LINFlex)	27
1.8.21	eTimer	28
1.8.22	Successive approximation Analog-to-Digital Converter (ADC)	29
1.8.23	Fault Collection Unit (FCU)	30
1.8.24	Cyclic Redundancy Check (CRC)	30
1.8.25	Video encoder	30
1.8.26	Serial Audio Interface (SAI)	30
1.8.27	Ethernet AVB (FEC + PTP + RTC)	31
1.8.27.1	Precision Time Protocol	31
1.8.27.2	RTC	32

Chapter 2 Memory Map

Chapter 3 Signal Description

3.1	Introduction	37
3.2	Signal Properties Summary	37
3.3	Supply pins	47
3.4	System pins	49
3.5	Pinouts	50

Chapter 4 Clock Architecture

4.1	Clock related modules	53
4.2	High-level block diagrams	53
4.3	Memory Map	59
4.4	Internal RC oscillator (IRC) digital interface	61
	4.4.1 Introduction	61
	4.4.2 Functional description	61
	4.4.3 Register description	62
4.5	External crystal oscillator (XOSC) digital interface	63
	4.5.1 Main features	63
	4.5.2 Functional description	63
	4.5.3 Register description	64
4.6	Frequency-modulated phase-locked loop (FMPLL)	65
	4.6.1 Introduction	65
	4.6.2 Overview	65
	4.6.3 Features	66
	4.6.4 Memory map	66
	4.6.5 Register description	67
	4.6.5.1 Control Register (CR)	67
	4.6.5.2 Modulation Register (MR)	69
	4.6.6 Functional description	70
	4.6.6.1 Normal mode	70
	4.6.6.2 Progressive clock switching	71
	4.6.6.3 Normal mode with frequency modulation	71
	4.6.6.4 Powerdown mode	72
	4.6.7 Recommendations	72
4.7	Clock Monitor Unit (CMU)	73
	4.7.1 Overview	73
	4.7.2 Main features	73
	4.7.3 Functional description	73
	4.7.4 Crystal clock monitor	74
	4.7.4.1 FMPLL clock monitor	74
	4.7.4.2 Frequency meter	74

4.7.5	Memory map and register description	75
4.7.5.1	Control status register (CMU_CSR)	76
4.7.5.2	Frequency display register (CMU_FDR)	77
4.7.5.3	High-frequency reference register A (CMU_HFREFR_A)	77
4.7.5.4	Low-frequency reference register A (CMU_LFREFR_A)	78
4.7.5.5	Interrupt status register (CMU_ISR)	78
4.7.5.6	Measurement duration register (CMU_MDR)	79
4.8	Boot and power management concept	79
4.9	Safety concept	81

Chapter 5 Clock Generation Module (MC_CGM)

5.1	Introduction	83
5.1.1	Overview	83
5.1.2	Features	84
5.2	External Signal Description	85
5.3	Memory Map and Register Definition	85
5.3.1	Register Descriptions	85
5.3.1.1	PLL Clock Divider Register (PLL_CLK_DIV)	86
5.3.1.2	System Clock Divider Register (SYSTEM_CLK_DIV)	86
5.3.1.3	RTC Clock Divider Register (RTC_CLK_DIV)	87
5.3.1.4	Output Clock Enable Register (CGM_OC_EN)	87
5.3.1.5	Output Clock Division Select Register (CGM_OCDS_SC)	88
5.3.1.6	System Clock Select Status Register (CGM_SC_SS)	89
5.4	Functional Description	89
5.4.1	System Clock Generation	89
5.4.1.1	System Clock Source Selection	90
5.4.1.2	System Clock Disable	90
5.4.2	Output Clock Multiplexing	90
5.4.3	Output Clock Division Selection	91

Chapter 6 Mode Entry Module (MC_ME)

6.1	Introduction	93
6.1.1	Overview	93
6.1.2	Features	95
6.1.3	Modes of Operation	95
6.2	External Signal Description	96
6.3	Memory Map and Register Definition	96
6.3.1	Memory Map	97
6.3.2	Register Description	105
6.3.2.1	Global Status Register (ME_GS)	105
6.3.2.2	Mode Control Register (ME_MCTL)	107
6.3.2.3	Mode Enable Register (ME_ME)	108
6.3.2.4	Interrupt Status Register (ME_IS)	110

6.3.2.5	Interrupt Mask Register (ME_IM)	111
6.3.2.6	Invalid Mode Transition Status Register (ME_IMTS)	112
6.3.2.7	Debug Mode Transition Status Register (ME_DMTS)	113
6.3.2.8	RESET Mode Configuration Register (ME_RESET_MC)	116
6.3.2.9	TEST Mode Configuration Register (ME_TEST_MC)	116
6.3.2.10	SAFE Mode Configuration Register (ME_SAFE_MC)	117
6.3.2.11	DRUN Mode Configuration Register (ME_DRUN_MC)	117
6.3.2.12	RUN0..3 Mode Configuration Register (ME_RUN0..3_MC)	118
6.3.2.13	HALT0 Mode Configuration Register (ME_HALT0_MC)	118
6.3.2.14	STOP0 Mode Configuration Register (ME_STOP0_MC)	119
6.3.2.15	Peripheral Status Register 0 (ME_PS0)	121
6.3.2.16	Peripheral Status Register 1 (ME_PS1)	121
6.3.2.17	Peripheral Status Register 2 (ME_PS2)	122
6.3.2.18	Peripheral Status Register 3 (ME_PS3)	122
6.3.2.19	Run Peripheral Configuration Registers (ME_RUN_PC0...7)	123
6.3.2.20	Low-Power Peripheral Configuration Registers (ME_LP_PC0...7)	124
6.3.2.21	Peripheral Control Registers (ME_PCTLn)	124
6.4	Functional Description	125
6.4.1	Mode Transition Request	125
6.4.2	Modes Details	127
6.4.2.1	RESET MODE	127
6.4.2.2	DRUN Mode	127
6.4.2.3	SAFE Mode	128
6.4.2.4	Test Mode	129
6.4.2.5	RUN0..3 Modes	129
6.4.2.6	HALT0 Mode	130
6.4.2.7	STOP0 Mode	130
6.4.3	Mode Transition Process	131
6.4.3.1	Target Mode Request	131
6.4.3.2	Target Mode Configuration Loading	132
6.4.3.3	Peripheral Clocks Disable	132
6.4.3.4	Processor Low-Power Mode Entry	133
6.4.3.5	Processor and System Memory Clock Disable	133
6.4.3.6	Clock Sources Switch-On	133
6.4.3.7	Flash Modules Switch-On	134
6.4.3.8	Pad Outputs-On	134
6.4.3.9	Peripheral Clocks Enable	134
6.4.3.10	Processor and Memory Clock Enable	134
6.4.3.11	Processor Low-Power Mode Exit	134
6.4.3.12	System Clock Switching	135
6.4.3.13	Pad Switch-Off	136
6.4.3.14	Clock Sources (with no Dependencies) Switch-Off	136
6.4.3.15	Clock Sources (with Dependencies) Switch-Off	136
6.4.3.16	Flash Switch-Off	136
6.4.3.17	Current Mode Update	136

6.4.4	Protection of Mode Configuration Registers	139
6.4.5	Mode Transition Interrupts	139
6.4.5.1	Invalid Mode Configuration Interrupt	139
6.4.5.2	Invalid Mode Transition Interrupt	140
6.4.5.3	SAFE Mode Transition Interrupt	141
6.4.5.4	Mode Transition Complete interrupt	141
6.4.6	Peripheral Clock Gating	141
6.4.7	Application Example	142

Chapter 7 Reset Generation Module (MC_RGM)

7.1	Introduction	145
7.1.1	Overview	145
7.1.2	Features	146
7.1.3	Reset Sources	147
7.2	External Signal Description	148
7.3	Memory Map and Register Definition	148
7.3.1	Register Descriptions	150
7.3.1.1	Functional Event Status Register (RGM_FES)	151
7.3.1.2	Destructive Event Status Register (RGM_DES)	152
7.3.1.3	Functional Event Reset Disable Register (RGM_FERD)	153
7.3.1.4	Functional Event Alternate Request Register (RGM_FEAR)	155
7.3.1.5	Functional Event Short Sequence Register (RGM_FESS)	156
7.3.1.6	Functional Bidirectional Reset Enable Register (RGM_FBRE)	157
7.4	Functional Description	158
7.4.1	Reset State Machine	158
7.4.1.1	PHASE0 Phase	160
7.4.1.2	PHASE1 Phase	161
7.4.1.3	PHASE2 Phase	161
7.4.1.4	PHASE3 Phase	161
7.4.1.5	IDLE Phase	161
7.4.2	Destructive Resets	161
7.4.3	External Reset	162
7.4.4	Functional Resets	162
7.4.5	Alternate Event Generation	163
7.4.6	Boot Mode Capturing	163

Chapter 8 Power Control Unit (MC_PCU)

8.1	Introduction	165
8.1.1	Overview	165
8.1.2	Features	166
8.2	External Signal Description	166
8.3	Memory Map and Register Definition	167
8.3.1	Memory Map	167

8.3.2	Register Descriptions	168
8.3.2.1	Power Domain Status Register (PCU_PSTAT)	168

Chapter 9 Power Management

9.1	Power management overview	169
9.1.1	Internal voltage regulation mode	169
9.1.2	External voltage regulation mode	170
9.1.3	Voltage Regulator Electrical Characteristics	171
9.2	Power sequencing	171
9.3	Power Management Unit (PMU)	172

Chapter 10 Interrupt Controller (INTC)

10.1	Introduction	173
10.2	Features	173
10.3	Block diagram	174
10.4	Modes of operation	175
10.4.1	Normal mode	175
10.4.1.1	Software vector mode	175
10.4.1.2	Hardware vector mode	176
10.4.1.3	Debug mode	176
10.4.1.4	Stop mode	176
10.5	Memory map and registers description	177
10.5.1	Module memory map	177
10.5.2	Registers description	177
10.5.2.1	INTC Module Configuration Register (INTC_MCR)	178
10.5.2.2	INTC Current Priority Register for Processor (INTC_CPR)	178
10.5.2.3	INTC Interrupt Acknowledge Register (INTC_IACKR)	180
10.5.2.4	INTC End-of-Interrupt Register (INTC_EOIR)	180
10.5.2.5	INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7)	181
10.5.2.6	INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR220_221)	182
10.6	Functional description	184
10.6.1	Interrupt request sources	193
10.6.1.1	Peripheral interrupt requests	193
10.6.1.2	Software configurable interrupt requests	193
10.6.1.3	Unique vector for each interrupt request source	193
10.6.2	Priority management	193
10.6.2.1	Current priority and preemption	193
10.6.2.1.1	Priority arbitrator subblock	194
10.6.2.1.2	Request selector subblock	194
10.6.2.1.3	Vector encoder subblock	194
10.6.2.1.4	Priority comparator subblock	194
10.6.2.2	Last-in first-out (LIFO)	194

10.6.3	Handshaking with processor	195
10.6.3.1	Software vector mode handshaking	195
10.6.3.1.1	Acknowledging interrupt request to processor	195
10.6.3.1.2	End of interrupt exception handler	195
10.6.3.2	Hardware vector mode handshaking	196
10.7	Initialization/application information	197
10.7.1	Initialization flow	197
10.7.2	Interrupt exception handler	197
10.7.2.1	Software vector mode	198
10.7.2.2	Hardware vector mode	198
10.7.3	ISR, RTOS, and task hierarchy	199
10.7.4	Order of execution	200
10.7.5	Priority ceiling protocol	201
10.7.5.1	Elevating priority	201
10.7.5.2	Ensuring coherency	201
10.7.6	Selecting priorities according to request rates and deadlines	201
10.7.7	Software configurable interrupt requests	202
10.7.7.1	Scheduling a lower priority portion of an ISR	202
10.7.7.2	Scheduling an ISR on another processor	203
10.7.8	Lowering priority within an ISR	203
10.7.9	Negating an interrupt request outside of its ISR	203
10.7.9.1	Negating an interrupt request as a side effect of an ISR	203
10.7.9.2	Negating multiple interrupt requests in one ISR	203
10.7.9.3	Proper setting of interrupt request priority	204
10.7.10	Examining LIFO contents	204

Chapter 11 Wakeup Unit (WKPU)

11.1	Introduction	205
11.1.1	Overview	205
11.1.2	Features	205
11.2	External signal description	206
11.3	Memory map and register description	206
11.3.1	Memory map	206
11.3.2	Register descriptions	206
11.3.2.1	NMI Status Flag Register (NSR)	206
11.3.2.2	NMI Configuration Register (NCR)	207
11.4	Functional description	209
11.4.1	General	209
11.4.2	Non-Maskable Interrupts	209
11.4.2.1	NMI management	210

Chapter 12 System Status and Configuration Module (SSCM)

12.1	Introduction	213
------	--------------------	-----

12.1.1	Overview	213
12.1.2	Features	213
12.1.3	Modes of Operation	214
12.2	External Signal Description	214
12.3	Memory Map/Register Definition	214
12.3.1	Register Descriptions	214
12.3.1.1	System Status Register	215
12.3.1.2	System Memory and ID Register	216
12.3.1.3	Error Configuration	217
12.3.1.4	Debug Status Port Register	218
12.3.1.5	Primary Boot Address	220
12.4	Functional Description	220
12.5	Initialization/Application Information	220
12.5.1	Reset	220

Chapter 13 System Integration Unit Lite (SIUL)

13.1	Introduction	221
13.2	Overview	221
13.3	Features	222
13.4	External signal description	223
13.4.1	Detailed signal descriptions	223
13.4.1.1	General-purpose I/O pins (GPIO[0:70])	223
13.4.1.2	External interrupt request input pins (EIRQ[0:31])	223
13.5	Memory map and register description	223
13.5.1	SIUL memory map	224
13.5.2	Register protection	225
13.5.3	Register description	225
13.5.3.1	MCU ID Register #1 (MIDR1)	225
13.5.3.2	MCU ID Register #2 (MIDR2)	227
13.5.3.3	Interrupt Status Flag Register (ISR)	228
13.5.3.4	Interrupt Request Enable Register (IRER)	228
13.5.3.5	Interrupt Rising-Edge Event Enable Register (IREER)	229
13.5.3.6	Interrupt Falling-Edge Event Enable Register (IFEER)	229
13.5.3.7	Interrupt Filter Enable Register (IFER)	230
13.5.3.8	Pad Configuration Registers (PCR[0:70])	230
13.5.3.9	Pad Selection for Multiplexed Inputs Registers (PSMI0–PSMI25)	232
13.5.3.10	GPIO Pad Data Output Registers (GPDO0_3–GPDO68_71)	235
13.5.3.11	GPIO Pad Data Input Registers (GPDI0_3–GPDI68_71)	236
13.5.3.12	Parallel GPIO Pad Data Out Registers (PGPDO0 – PGPDO2)	237
13.5.3.13	Parallel GPIO Pad Data In Register (PGPDI0 – PGPDI2)	237
13.5.3.14	Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO4)	238
13.5.3.15	Interrupt Filter Maximum Counter Registers (IFMC0–IFMC31)	239
13.5.3.16	Interrupt Filter Clock Prescaler Register (IFCPR)	240
13.6	Functional description	241

13.6.1	Pad control	241
13.6.2	General purpose input and output pads (GPIO)	241
13.6.3	External interrupts	242
13.7	Pin muxing	243

Chapter 14 e200z0h Core

14.1	Overview	245
14.2	Features	245
14.2.1	Microarchitecture summary	246
14.2.1.1	Block diagram	247
14.2.1.2	Instruction unit features	248
14.2.1.3	Integer unit features	248
14.2.1.4	Load/Store unit features	248
14.2.1.5	e200z0h system bus features	248
14.2.1.6	Nexus features	249
14.3	Core registers and programmer's model	249
14.3.1	Unimplemented SPRs and read-only SPRs	252
14.4	Instruction summary	252

Chapter 15 Crossbar Switch (XBAR)

15.1	Introduction	253
15.2	Block diagram	253
15.3	Overview	254
15.4	Features	254
15.5	Modes of operation	255
15.5.1	Normal mode	255
15.5.2	Debug mode	255
15.6	Functional description	255
15.6.1	Overview	255
15.6.2	General operation	255
15.6.3	Master ports	256
15.6.4	Slave ports	256
15.6.5	Priority assignment	256
15.6.6	Arbitration	257
15.6.6.1	Fixed priority operation	257
15.6.6.1.1	Parking	257

Chapter 16 Miscellaneous Control Module (MCM)

16.1	Introduction	259
16.2	Overview	259
16.3	Features	259
16.4	Memory Map and Registers Description	259

16.4.1	Memory Map	260
16.4.2	Registers Description	261
16.4.2.1	Processor Core Type (PCT) register	261
16.4.2.2	Revision (REV) register	261
16.4.2.3	IPS Module Configuration (IMC) register	262
16.4.2.4	Miscellaneous Interrupt Register (MIR)	263
16.4.2.5	Miscellaneous User-Defined Control Register (MUDCR)	263
16.4.2.6	ECC registers	264
16.4.2.7	ECC Configuration Register (ECR)	265
16.4.2.8	ECC Status Register (ESR)	266
16.4.2.9	ECC Error Generation Register (EEGR)	268
16.4.2.10	Flash ECC Address Register (FEAR)	270
16.4.2.11	Flash ECC Master Number Register (FEMR)	271
16.4.2.12	Flash ECC Attributes (FEAT) register	272
16.4.2.13	Flash ECC Data Register (FEDR)	272
16.4.2.14	RAM ECC Address Register (REAR)	273
16.4.2.15	RAM ECC Syndrome Register (RESR)	274
16.4.2.16	RAM ECC Master Number Register (REMR)	276
16.4.2.17	RAM ECC Attributes (REAT) register	277
16.4.2.18	RAM ECC Data Register (REDR)	277
16.4.3	MCM_reg_protection	278

Chapter 17 Internal Static RAM (SRAM)

17.1	Introduction	281
17.2	SRAM operating mode	281
17.3	Register memory map	281
17.4	SRAM ECC mechanism	281
17.4.1	Access timing	282
17.4.2	Reset effects on SRAM accesses	283
17.5	Functional description	283
17.6	Initialization and application information	283

Chapter 18 Flash Memory

18.1	Introduction	285
18.2	Platform flash controller	286
18.2.1	Introduction	286
18.2.1.1	Overview	286
18.2.1.2	Features	287
18.2.2	Modes of operation	287
18.2.3	External signal descriptions	287
18.2.4	Memory map and registers description	288
18.2.4.1	Memory map	288
18.2.4.2	Registers description	290

18.2.4.2.1	Platform Flash Configuration Register 0 (PFCR0)	290
18.2.4.2.2	Platform Flash Configuration Register 1 (PFCR1)	293
18.2.4.2.3	Platform Flash Access Protection Register (PFAPR)	295
18.2.5	Functional description	296
18.2.6	Basic interface protocol	297
18.2.7	Access protections	297
18.2.8	Read cycles — buffer miss	297
18.2.9	Read cycles — buffer hit	298
18.2.10	Write cycles	298
18.2.11	Error termination	298
18.2.12	Access pipelining	299
18.2.13	Flash error response operation	299
18.2.14	Bank0 page read buffers and prefetch operation	299
18.2.14.1	Instruction/data prefetch triggering	301
18.2.14.2	Per-master prefetch triggering	301
18.2.14.3	Buffer allocation	301
18.2.14.4	Buffer invalidation	301
18.2.15	Bank1 temporary holding register	302
18.2.16	Read-While-Write functionality	302
18.2.17	Wait state emulation	304
18.2.18	Timing diagrams	305
18.3	Code Flash Memory (C90LC)	311
18.3.1	Overview	311
18.3.2	Features	311
18.3.3	Block Diagram	311
18.3.4	Functional Description	312
18.3.4.1	Macrocell Structure	312
18.3.5	Code flash sectorization	313
18.3.5.1	Test Flash Block	314
18.3.5.2	Shadow block	315
18.3.5.3	User Mode Operation	316
18.3.5.4	Reset	316
18.3.5.5	Disable Mode (Power-Down)	317
18.3.5.6	Sleep Mode (Low Power Mode)	317
18.3.6	Registers Description	318
18.3.6.1	Module Configuration Register (MCR)	319
18.3.6.2	Low/Mid Address Space Block Locking register (LML)	324
18.3.6.3	Non-Volatile Low/Mid Address Space Block Locking register (NVLML)	324
18.3.6.4	High address space Block Locking register (HBL)	326
18.3.6.5	Non Volatile High address space Block Locking register (NVHBL)	327
18.3.6.6	Secondary Low/Mid Address Space Block Locking register (SLL)	327
18.3.6.7	Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL)	328
18.3.6.8	Low/Mid Address Space Block Select register (LMS)	330
18.3.6.9	High address space Block Select register (HBS)	331

18.3.6.10	Address Register (ADR)	332
18.3.6.11	Bus Interface Unit 0 register (BIU0)	333
18.3.6.12	Bus Interface Unit 1 register (BIU1)	333
18.3.6.13	Bus Interface Unit 2 register (BIU2)	334
18.3.6.13.1	Non-volatile Bus Interface Unit 2 register (NVBIU2)	334
18.3.6.14	Non Volatile Bus Interface Unit 3 register (NVBIU3)	335
18.3.6.15	User Test 0 register (UT0)	335
18.3.6.16	User Test 1 register (UT1)	337
18.3.6.17	User Test 2 register (UT2)	338
18.3.6.18	User Multiple Input Signature Register 0 (UMISR0)	339
18.3.6.19	User Multiple Input Signature Register 1 (UMISR1)	339
18.3.6.20	User Multiple Input Signature Register 2 (UMISR2)	340
18.3.6.21	User Multiple Input Signature Register 3 (UMISR3)	341
18.3.6.22	User Multiple Input Signature Register 4 (UMISR4)	341
18.3.6.23	Non-Volatile Private Censorship Password 0 register (NVPWD0)	342
18.3.6.24	Non-Volatile Private Censorship Password 1 register (NVPWD1)	343
18.3.6.25	Non-Volatile System Censoring Information 0 register (NVSCIO)	343
18.3.6.26	Non-Volatile System Censoring Information 1 register (NVSCI1)	344
18.3.6.27	Non-Volatile User Options register (NVUSRO)	345
18.3.7	Programming Considerations	346
18.3.7.1	Modify Operations	346
18.3.7.1.1	Double Word Program	347
18.3.7.1.2	Block Erase	349
18.3.7.1.3	Erase Suspend/Resume	350
18.3.7.1.4	User Test Mode	350
18.3.7.2	Error correction code	354
18.3.7.2.1	ECC algorithms	355
18.3.7.3	EEProm emulation	355
18.3.7.4	Eprom Emulation	355
18.3.7.4.1	All '1's No Error	355
18.3.7.5	Protection strategy	356
18.3.7.5.1	Modify protection	356
18.3.7.5.2	Censored Mode	356
18.4	Data Flash Memory	357
18.4.1	Block Overview	357
18.4.2	Features	358
18.4.3	Block Diagram	358
18.4.4	Functional Description	359
18.4.4.1	Macrocell Structure	359
18.4.4.2	Data flash sectorization	359
18.4.4.3	Test Flash Block	360
18.4.4.4	Reset	361
18.4.4.5	Power-down mode	361
18.4.4.6	Slave Mode	362
18.4.5	Register description	362

18.4.5.1	Module Configuration Register (MCR)	363
18.4.5.2	Low/Mid Address Space Block Locking register (LML)	367
18.4.5.3	Non-Volatile Low/Mid Address Space Block Locking register (NVLML)	368
18.4.5.4	Secondary Low/Mid Address Space Block Locking register (SLL)	369
18.4.5.5	Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL)	370
18.4.5.6	Low/Mid Address Space Block Select register (LMS)	371
18.4.5.7	Address Register (ADR)	372
18.4.5.8	User Test 0 register (UT0)	373
18.4.5.9	User Test 1 register (UT1)	375
18.4.5.10	User Multiple Input Signature Register 0 (UMISR0)	376
18.4.5.11	User Multiple Input Signature Register 1 (UMISR1)	376
18.4.6	Programming considerations	377
18.4.6.1	Modify operation	377
18.4.6.2	Word program	378
18.4.6.3	Sector erase	379
18.4.6.3.1	Erase suspend/resume	380
18.4.6.4	User Test Mode	381
18.4.6.4.1	Array integrity self check	381
18.4.6.4.2	Margin read	382
18.4.6.4.3	ECC logic check	383
18.4.7	Error correction code	384
18.4.7.1	ECC algorithms	384
18.4.7.2	ECC Algorithms Features	384
18.4.8	Protection strategy	385
18.4.8.1	Modify protection	385

Chapter 19 Enhanced Direct Memory Access (eDMA)

19.1	Introduction	387
19.1.1	Overview	387
19.1.2	Features	388
19.2	Memory map/register definition	393
19.2.1	Register descriptions	394
19.2.1.1	eDMA Control Register (DMACR)	394
19.2.1.2	eDMA Error Status (DMAES)	397
19.2.1.3	eDMA Enable Request (DMAERQL)	400
19.2.1.4	eDMA Enable Error Interrupt (DMAEEIL)	400
19.2.1.5	eDMA Set Enable Request (DMASERQ)	401
19.2.1.6	eDMA Clear Enable Request (DMACERQ)	402
19.2.1.7	eDMA Set Enable Error Interrupt (DMASEEI)	402
19.2.1.8	eDMA Clear Enable Error Interrupt (DMACEEI)	403
19.2.1.9	eDMA Clear Interrupt Request (DMACINT)	403
19.2.1.10	eDMA Clear Error (DMACERR)	404
19.2.1.11	eDMA Set START Bit (DMASSRT)	405

19.2.1.12 eDMA Clear DONE Status (DMACDNE)	405
19.2.1.13 eDMA Interrupt Request (DMAINTL)	406
19.2.1.14 eDMA Error (DMAERRL)	407
19.2.1.15 eDMA Hardware Request Status (DMAHRSL)	407
19.2.1.16 eDMA Channel n Priority (DCHPRIn), n = 0,..., {15}	408
19.2.1.17 Transfer Control Descriptor (TCD)	409
19.3 Functional description	419
19.3.1 eDMA microarchitecture	419
19.3.2 eDMA basic data flow	420
19.4 Initialization/application information	423
19.4.1 eDMA initialization	423
19.4.2 eDMA programming errors	424
19.4.3 eDMA arbitration mode considerations	425
19.4.3.1 Fixed group arbitration, fixed channel arbitration	425
19.4.3.2 Round-robin group arbitration, fixed channel arbitration	425
19.4.3.3 Round-robin group arbitration, round-robin channel arbitration	425
19.4.3.4 Fixed group arbitration, round-robin channel arbitration	426
19.4.4 eDMA transfer	426
19.4.4.1 Single request	426
19.4.4.2 Multiple requests	427
19.4.5 TCD status	429
19.4.5.1 Minor loop complete	429
19.4.5.2 Active channel TCD reads	429
19.4.5.3 Preemption status	430
19.4.6 Channel linking	430
19.4.7 Dynamic programming	431
19.4.7.1 Dynamic priority changing	431
19.4.7.2 Dynamic channel linking	431
19.4.7.3 Dynamic scatter/gather	432
19.4.7.3.1 Method 1 (channel not using major loop channel linking)	432
19.4.7.3.2 Method 2 (channel using major loop linking)	433
19.4.8 Hardware request release timing	434

Chapter 20 DMACHMUX

20.1 Introduction	435
20.1.1 Overview	435
20.1.2 Features	435
20.1.3 Modes of Operation	436
20.2 External Signal Description	436
20.2.1 Overview	436
20.3 Memory Map and Register Definition	436
20.3.1 Register Descriptions	437
20.3.1.1 Channel Configuration Registers	437
20.4 DMA request mapping	438

20.5	Functional Description	440
20.5.1	DMA Channels with periodic triggering capability	440
20.5.2	DMA Channels with no triggering capability	442
20.5.3	"Always Enabled" DMA Sources	442
20.6	Initialization/Application Information	443
20.6.1	Reset	443
20.6.2	Enabling and Configuring Sources	443
20.6.3	Freezing in STOP and HALT mode	446

Chapter 21 Video Encoder Wrapper

21.1	Introduction	447
21.1.1	Features	448
21.2	Block Diagram	449
21.2.1	MJPEG Video Encoder	449
21.2.2	MJPEG Operation Modes	450
21.2.2.1	Configuration Mode	451
21.2.2.2	Encoding Mode	453
21.2.2.3	Rate Control operation	454
21.3	Memory Map and Register Definition	455
21.3.1	Memory Map	455
21.3.2	Register Descriptions	457
21.3.2.1	Status_config	457
21.3.2.2	Picture_size	459
21.3.2.3	Pixel count	460
21.3.2.4	Dma_address	460
21.3.2.5	Dma_vstart_address	461
21.3.2.6	Dma_vend_address	461
21.3.2.7	Dma_alarm_address	462
21.3.2.8	Subchannel buffer start	462
21.3.2.9	JPEG In Offset Address	463
21.3.2.10	RC_REGS_SEL	463
21.3.2.11	LUMTH	464
21.3.2.12	MODE	465
21.3.2.13	CFG_MODE	466
21.3.2.14	CHRTH	467
21.3.2.15	Status registers	468
21.3.2.16	JPEG Stat 0	468
21.3.2.17	JPEG Stat 1	468
21.3.2.18	JPEG Stat 2	469
21.3.2.19	JPEG Stat 3	469
21.3.2.20	JPEG Stat 4	470
21.3.2.21	JPEG Stat 5	470
21.3.2.22	JPEG Stat 6	471
21.3.2.23	JPEG Stat 7	471

21.3.2.24 JPEG Stat 8	472
21.3.2.25 JPEG Stat 9	472
21.3.2.26 JPEG Stat 10	473
21.3.2.27 JPEG Stat 11	473
21.3.2.28 JPEG Stat 12	474
21.3.2.29 JPEG Stat 13	475
21.3.2.30 JPEG Stat 14	475
21.3.2.31 JPEG Stat 15	476
21.4 Functional Description	476
21.4.1 Input interface	477
21.4.1.1 External Sync Interface Timing Diagram	477
21.4.1.2 ITU-BT656 sync information extraction	478
21.4.1.3 Video In Data Format for embedded Sync Mode	480
21.4.1.4 Video In Format for External Sync Mode	481
21.4.2 Circular buffer	482
21.4.3 Subchannel Mode	483
21.4.4 Programming Sequence	485

Chapter 22

Integrated Interchip Sound (I²S) / Synchronous Audio Interface (SAI)

22.1 Introduction	487
22.1.1 Features	487
22.1.2 Modes of Operation	487
22.1.2.1 Run Mode	487
22.1.2.2 Debug Mode	487
22.2 External signals	488
22.3 Memory Map and Registers	488
22.3.1 SAI Transmit Control Register (I2S_TCSR)	490
22.3.2 SAI Transmit Configuration 1 Register (I2S_TCR1)	490
22.3.3 SAI Transmit Configuration 4 Register (I2S_TCR4)	493
22.3.4 SAI Transmit Configuration 5 Register (I2S_TCR5)	494
22.3.5 SAI Transmit Data Register (I2S_TDR)	495
22.3.6 SAI Transmit FIFO Register (I2S_TFR)	495
22.3.7 SAI Transmit Mask Register (I2S_TMR)	496
22.3.8 SAI Receive Control Register (I2S_RCSR)	496
22.3.9 SAI Receive Configuration 1 Register (I2S_RCR1)	499
22.3.10 SAI Receive Configuration 2 Register (I2S_RCR2)	499
22.3.11 SAI clocking	504
22.3.11.1 Audio Master Clock	504
22.3.11.2 Bit Clock	505
22.3.11.3 Bus Clock	505
22.3.12 SAI resets	505
22.3.12.1 Software reset	505
22.3.12.2 FIFO reset	506
22.3.13 Synchronous Modes	506

22.3.13.1 Synchronous Mode	506
22.3.13.2 Multiple SAI Synchronous Mode	506
22.3.14 Frame sync configuration	507
22.3.15 Data FIFO	507
22.3.15.1 Data alignment	507
22.3.15.2 FIFO pointers	508
22.3.16 Word mask register	509
22.3.17 Interrupts and DMA requests	509
22.3.17.1 FIFO data ready flag	509
22.3.17.2 FIFO warning flag	509
22.3.17.3 FIFO error flag	509
22.3.17.4 Sync error flag	510
22.3.17.5 Word start flag	510

Chapter 23 SAI Instantiation

23.1 Introduction	511
23.1.1 SAI/I2S Overview	511
23.1.2 External Signals Multiplexing	512
23.1.3 SAI/I2S Clocking	512
23.1.3.1 SAI/I2S Clock Selection	512
23.1.3.2 CLKMODE in SAI/I2S TCR2 Register	512
23.1.3.3 CLKMODE in SAI/I2S RCR2 Register	513
23.1.3.4 Configuring clock source for SAI/I2S audio master clock	514

Chapter 24 Deserial Serial Peripheral Interface (DSPI)

24.1 Introduction	515
24.2 Block diagram	515
24.3 Overview	516
24.4 Features	516
24.5 Modes of operation	517
24.5.1 Master mode	517
24.5.2 Slave mode	517
24.5.3 Module disable mode	518
24.5.4 Debug mode	518
24.6 External signal description	518
24.6.1 Signal overview	518
24.6.2 Signal names and descriptions	519
24.6.2.1 Peripheral Chip Select / Slave Select (CS ₀)	519
24.6.2.2 Peripheral Chip Selects 1–3 ($\overline{\text{CS}}1:3$)	519
24.6.2.3 Peripheral Chip Select 4 / Master Trigger ($\overline{\text{CS}}4/\text{MTRIG}$)	519
24.6.2.4 Peripheral Chip Select 5/Peripheral Chip Select Strobe (CS ₅)	519
24.6.2.5 Serial Input (SIN _x)	519
24.6.2.6 Serial Output (SOUT _x)	519

24.6.2.7	Serial Clock (SCK _x)	520
24.7	Memory map and registers description	520
24.7.1	Memory map	520
24.7.2	Registers description	521
24.7.2.1	DSPI Module Configuration Register (DSPIx_MCR)	521
24.7.2.2	DSPI Hardware Configuration Register (DSPI_HCR)	524
24.7.2.3	DSPI Transfer Count Register (DSPIx_TCR)	525
24.7.2.4	DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn)	525
24.7.2.5	DSPI Status Register (DSPIx_SR)	531
24.7.2.6	DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER)	533
24.7.2.7	DSPI PUSH TX FIFO Register (DSPIx_PUSHR)	534
24.7.2.8	DSPI POP RX FIFO Register (DSPIx_POPR)	536
24.7.2.9	DSPI Transmit FIFO Registers 0–4 (DSPIx_TXFRn)	537
24.7.2.10	DSPI Receive FIFO Registers 0–4 (DSPIx_RXFRn)	537
24.8	Functional description	538
24.8.1	Modes of operation	539
24.8.1.1	Master mode	539
24.8.1.2	Slave mode	540
24.8.1.3	Module disable mode	540
24.8.1.4	Debug mode	540
24.8.2	Start and stop of DSPI transfers	540
24.8.3	Serial Peripheral Interface (SPI) configuration	541
24.8.3.1	SPI master mode	541
24.8.3.2	SPI slave mode	542
24.8.3.3	FIFO disable operation	542
24.8.3.4	Transmit First In First Out (TX FIFO) buffering mechanism	542
24.8.3.4.1	Filling the TX FIFO	543
24.8.3.4.2	Draining the TX FIFO	543
24.8.3.5	Receive First In First Out (RX FIFO) buffering mechanism	543
24.8.3.5.1	Filling the RX FIFO	544
24.8.3.5.2	Draining the RX FIFO	544
24.8.4	DSPI baud rate and clock delay generation	544
24.8.4.1	Baud rate generator	544
24.8.4.2	CS to SCK delay (t _{CSC})	545
24.8.4.3	After SCK delay (t _{ASC})	545
24.8.4.4	Delay after transfer (t _{DT})	546
24.8.4.5	Peripheral Chip Select strobe enable ($\overline{CS5_x}$)	546
24.8.5	Transfer formats	547
24.8.5.1	Classic SPI transfer format (CPHA = 0)	548
24.8.5.2	Classic SPI transfer format (CPHA = 1)	549
24.8.5.3	Modified SPI transfer format (MTFE = 1, CPHA = 0)	549
24.8.5.4	Modified SPI transfer format (MTFE = 1, CPHA = 1)	551
24.8.5.5	Continuous selection format	551
24.8.5.6	Clock polarity switching between DSPI transfers	553

24.8.5.7	Fast Continuous Selection Format	553
24.8.6	Continuous Serial communications clock	555
24.8.7	Interrupts/DMA requests	556
24.8.7.1	End of queue interrupt request (EOQF)	556
24.8.7.2	Transmit FIFO fill interrupt or DMA request (TFFF)	557
24.8.7.3	Transfer complete interrupt request (TCF)	557
24.8.7.4	Transmit FIFO underflow interrupt request (TFUF)	557
24.8.7.5	Receive FIFO drain interrupt or DMA request (RFDF)	557
24.8.7.6	Receive FIFO overflow interrupt request (RFOF)	557
24.8.7.7	FIFO overrun request (TFUF) or (RFOF)	557
24.8.8	Power saving features	558
24.8.8.1	Module disable mode	558
24.9	Initialization and application information	558
24.9.1	Managing queues	558
24.9.2	Baud rate settings	559
24.9.3	Delay settings	560
24.9.4	MPC5604E DSPI compatibility with QSPI of the MPC500 MCUs	560
24.9.5	Calculation of FIFO pointer addresses	561
24.9.5.1	Address calculation for first-in entry and last-in entry in TX FIFO	562
24.9.5.2	Address calculation for first-in entry and last-in entry in RX FIFO	562

Chapter 25 LIN Controller (LINFlex)

25.1	Introduction	565
25.2	Main features	565
25.3	General description	566
25.4	Fractional baud rate generation	567
25.5	Operating modes	570
25.5.1	Initialization mode	570
25.5.2	Normal mode	570
25.5.3	Low power mode (Sleep)	571
25.6	Test modes	571
25.6.1	Loop Back mode	571
25.6.2	Self Test mode	571
25.7	Memory map and registers description	572
25.7.1	Memory map	572
25.7.2	Register description	574
25.7.2.1	LIN control register 1 (LINCR1)	575
25.7.2.2	LIN interrupt enable register (LINIER)	578
25.7.2.3	LIN status register (LINSR)	579
25.7.2.4	LIN error status register (LINESR)	582
25.7.2.5	UART mode control register (UARTCR)	583
25.7.2.6	UART mode status register (UARTSR)	585
25.7.2.7	LIN timeout control status register (LINTCSR)	587
25.7.2.8	LIN output compare register (LINOCR)	588

25.7.2.9	LIN timeout control register (LINTOCR)	588
25.7.2.10	LIN fractional baud rate register (LINFBR)	589
25.7.2.11	LIN Integer Baud Rate Register (LINIBRR)	590
25.7.2.12	LIN checksum field register (LINCFR)	590
25.7.2.13	LIN control register 2 (LINCR2)	591
25.7.2.14	Buffer identifier register (BIDR)	592
25.7.2.15	Buffer data register least significant (BDRL)	593
25.7.2.16	Buffer data register most significant (BDRM)	594
25.7.2.17	Identifier filter enable register (IFER)	594
25.7.2.18	Identifier filter match index (IFMI)	596
25.7.2.19	Identifier filter mode register (IFMR)	596
25.7.2.20	Identifier filter control register (IFCR2 _n)	598
25.7.2.21	Identifier filter control register (IFCR2 _n + 1)	599
25.8	Functional description	600
25.8.1	UART mode	600
25.8.1.1	Buffer in UART mode	600
25.8.1.2	UART transmitter	601
25.8.1.3	UART receiver	601
25.8.1.4	Clock gating	602
25.8.2	LIN mode	602
25.8.2.1	Master mode	602
25.8.2.1.1	LIN header transmission	602
25.8.2.1.2	Data transmission (transceiver as publisher)	602
25.8.2.1.3	Data reception (transceiver as subscriber)	603
25.8.2.1.4	Data discard	603
25.8.2.1.5	Error detection	603
25.8.2.1.6	Error handling	603
25.8.2.2	Slave mode	604
25.8.2.2.1	Data transmission (transceiver as publisher)	604
25.8.2.2.2	Data reception (transceiver as subscriber)	604
25.8.2.2.3	Data discard	605
25.8.2.2.4	Error detection	605
25.8.2.2.5	Error handling	605
25.8.2.2.6	Valid header	606
25.8.2.2.7	Valid message	606
25.8.2.2.8	Overrun	606
25.8.2.3	Slave mode with identifier filtering	606
25.8.2.3.1	Filter mode	606
25.8.2.3.2	Identifier filter mode configuration	607
25.8.2.4	Slave mode with automatic resynchronization	608
25.8.2.4.1	Deviation error on the Synch Field	609
25.8.2.5	Clock gating	610
25.8.3	8-bit timeout counter	610
25.8.3.1	LIN timeout mode	610
25.8.3.1.1	LIN Master mode	610

25.8.3.1.2 LIN Slave mode	611
25.8.3.2 Output compare mode	611
25.8.4 Interrupts	612

Chapter 26 FlexCAN Module

26.1 Introduction	613
26.1.1 Overview	614
26.1.2 FlexCAN Module Features	615
26.1.3 Modes of Operation	616
26.2 External Signal Description	617
26.2.1 Overview	617
26.2.2 Signal Descriptions	617
26.2.2.1 CAN Rx	617
26.2.2.2 CAN Tx	617
26.3 Memory Map/Register Definition	617
26.3.1 FlexCAN Memory Mapping	617
26.3.2 Message Buffer Structure	618
26.3.3 Rx FIFO Structure	622
26.3.4 Register Descriptions	623
26.3.4.1 Module Configuration Register (MCR)	624
26.3.4.2 Control Register (CTRL)	628
26.3.4.3 Free Running Timer (TIMER)	632
26.3.4.4 Rx Global Mask (RXGMASK)	633
26.3.4.5 Rx 14 Mask (RX14MASK)	634
26.3.4.6 Rx 15 Mask (RX15MASK)	634
26.3.4.7 Error Counter Register (ECR)	634
26.3.4.8 Error and Status Register (ESR)	636
26.3.4.9 Interrupt Masks 1 Register (IMASK1)	639
26.3.4.10 Interrupt Flags 1 Register (IFLAG1)	640
26.3.4.11 Rx Individual Mask Registers (RXIMR0–RXIMR31)	641
26.4 Functional Description	642
26.4.1 Overview	642
26.4.2 Transmit Process	642
26.4.3 Arbitration process	643
26.4.4 Receive Process	643
26.4.5 Matching Process	645
26.4.6 Data Coherence	646
26.4.6.1 Message Buffer Deactivation	646
26.4.6.2 Message Buffer Lock Mechanism	647
26.4.7 Rx FIFO	648
26.4.8 CAN Protocol Related Features	649
26.4.8.1 Remote Frames	649
26.4.8.2 Overload Frames	649
26.4.8.3 Time Stamp	649

26.4.8.4	Protocol Timing	650
26.4.8.5	Arbitration and Matching Timing	652
26.4.9	Modes of Operation Details	653
26.4.9.1	Freeze Mode	653
26.4.9.2	Module Disable Mode	653
26.4.9.3	Stop Mode	654
26.4.10	Interrupts	655
26.4.11	Bus Interface	655
26.5	Initialization/Application Information	656
26.5.1	FlexCAN Initialization Sequence	656
26.5.2	FlexCAN Addressing and RAM size configurations	657

Chapter 27 Analog-to-Digital Converter (ADC)

27.1	Overview	659
27.2	Introduction	659
27.2.1	Features	659
27.2.2	Block Diagram	660
27.3	Register descriptions	661
27.3.1	Introduction	661
27.3.2	Control logic registers	662
27.3.2.1	Main Configuration Register (MCR)	662
27.3.2.2	Main Status Register (MSR)	665
27.3.3	Interrupt registers	666
27.3.3.1	Interrupt Status Register (ISR)	666
27.3.3.2	Interrupt Mask Register (IMR)	667
27.3.3.3	Watchdog Threshold Interrupt Status Register (WTISR)	669
27.3.3.4	Watchdog Threshold Interrupt Mask Register (WTIMR)	670
27.3.4	DMA registers	671
27.3.4.1	DMA Enable Register (DMAE)	671
27.3.4.2	DMA Channel Select Register 0 (DMAR0)	672
27.3.5	Threshold registers	672
27.3.5.1	Introduction	672
27.3.5.2	Threshold Register (THRHLR[0:3])	672
27.3.6	Conversion timing registers CTR[0..1]	673
27.3.7	Mask registers	676
27.3.7.1	Introduction	676
27.3.7.2	Normal Conversion Mask Register 0 (NCMR0)	676
27.3.7.3	Injected Conversion Mask Register 0 (JCMR0)	677
27.3.8	Power Down Exit Delay Register (PDEDL)	677
27.3.9	Data registers	678
27.3.9.1	Introduction	678
27.3.9.2	Channel Data Register (CDR[0..6])	678
27.3.9.3	Channel Watchdog Select Register (CWSELR0)	679
27.3.9.4	Channel Watchdog Enable Register (CWENR0)	680

27.4	Functional description	680
27.4.1	Analog channel conversion	680
27.4.1.1	Normal conversion	680
27.4.1.2	Start of normal conversion	680
27.4.1.3	Normal conversion operating modes	681
27.4.1.4	Injected channel conversion	682
27.4.2	Analog clock generator and conversion timings	683
27.4.3	ADC sampling and conversion timing	683
27.4.4	Programmable analog watchdog	685
27.4.4.1	Introduction	685
27.4.5	DMA functionality	686
27.4.6	Interrupts	686
27.4.7	Power-down mode	687
27.4.8	Auto-clock-off mode	687

Chapter 28

Enhanced Motor Control Timer (eTimer)

28.1	Introduction	689
28.1.1	Overview	689
28.1.2	Features	689
28.1.3	Customization	690
28.1.4	Module Block Diagram	690
28.1.5	Channel Block Diagram	691
28.2	External Signal Descriptions	692
28.2.1	TIO[5:0] - Timer Input/Outputs	692
28.2.2	TAI[2] - Timer Auxiliary Input	692
28.3	Functional Description	692
28.3.1	General	692
28.3.2	Counting Modes	693
28.3.2.1	STOP Mode	693
28.3.2.2	COUNT Mode	694
28.3.2.3	EDGE-COUNT Mode	694
28.3.2.4	GATED-COUNT Mode	694
28.3.2.5	QUADRATURE-COUNT Mode	694
28.3.2.6	SIGNED-COUNT Mode	694
28.3.2.7	TRIGGERED-COUNT Mode	695
28.3.2.8	ONE-SHOT Mode	695
28.3.2.9	CASCADE-COUNT Mode	695
28.3.2.10	PULSE-OUTPUT Mode	696
28.3.2.11	FIXED-FREQUENCY PWM Mode	696
28.3.2.12	VARIABLE-FREQUENCY PWM Mode	696
28.3.2.13	Usage of Compare Registers	698
28.3.2.14	Usage of Compare Load Registers	699
28.3.2.15	MODULO COUNTING Mode	699
28.3.2.16	Compare Register and OFLAG Operation	700

28.3.3	Other Features	701
28.3.3.1	Redundant OFLAG Checking	701
28.3.3.2	Loopback Checking	701
28.3.3.3	Input Capture Mode	701
28.3.3.4	Master/Slave Mode	701
28.3.3.5	Watchdog Timer	701
28.4	Memory Map and Registers	702
28.4.1	Overview	702
28.4.2	Module Memory Map	702
28.4.3	Register Descriptions	703
28.4.4	Timer Channel Registers	703
28.4.4.1	Compare Register 1 (COMP1)	703
28.4.4.2	Compare Register 2 (COMP2)	703
28.4.4.3	Capture Register 1 (CAPT1)	704
28.4.4.4	Capture Register 2 (CAPT2)	704
28.4.4.5	Load Register (LOAD)	704
28.4.4.6	Hold Register (HOLD)	705
28.4.4.7	Counter Register (CNTR)	705
28.4.4.8	Control Register 1 (CTRL1)	705
28.4.4.9	Control Register 2 (CTRL2)	709
28.4.4.10	Control Register 3 (CTRL3)	711
28.4.4.11	Status Register (STS)	712
28.4.4.12	Interrupt and DMA Enable Register (INTDMA)	714
28.4.4.13	Comparator Load Register 1 (CMPLD1)	715
28.4.4.14	Comparator Load Register 2 (CMPLD2)	715
28.4.4.15	Compare and Capture Control Register (CCCTRL)	715
28.4.4.16	Input Filter Register (FILT)	718
28.4.4.16.1	Input Filter Considerations	718
28.4.5	Watchdog Timer Registers	718
28.4.5.1	Watchdog Time-out Registers (WDTOL and WDTOH)	719
28.4.6	Configuration Registers	719
28.4.6.1	Channel Enable Register (ENBL)	719
28.4.6.2	DMA Request Select Registers (DREQ0, DREQ1)	720
28.5	Resets	721
28.6	Clocks	721
28.7	Interrupts	722
28.8	DMA	722
28.9	ADC Trigger	723

Chapter 29 Fault Collection Unit (FCU)

29.1	Introduction	725
29.1.1	Overview	725
29.1.1.1	General description	725
29.1.2	Features	728

29.1.3	Modes of operation	728
29.1.3.1	Normal mode	728
29.1.3.2	Test mode	728
29.2	Memory map and register definition	728
29.2.1	Memory map	729
29.2.2	Register summary	729
29.2.3	Register descriptions	731
29.2.3.1	Module Configuration Register (FCU_MCR)	731
29.2.3.2	Fault Flag Register (FCU_FFR)	732
29.2.3.3	Frozen Fault Flag Register (FCU_FFFR)	734
29.2.3.4	Fake Fault Generation Register (FCU_FFGR)	735
29.2.3.5	Fault Enable Register (FCU_FER)	736
29.2.3.6	Key Register (FCU_KR)	736
29.2.3.7	Timeout Register (FCU_TR)	737
29.2.3.8	Timeout Enable Register (FCU_TER)	738
29.2.3.9	Module State Register (FCU_MSR)	738
29.2.3.10	Microcontroller State Register (FCU_MCSR)	739
29.2.3.11	Frozen MC State Register (FCU_FMCSR)	740
29.3	Functional description	741
29.3.1	State machine	742
29.3.2	Output generation protocol	743
29.3.2.1	Dual-rail protocol	744
29.3.2.2	Time switching protocol	745
29.3.2.3	Bi-Stable protocol	745

Chapter 30 Periodic Interrupt Timer (PIT_RTI)

30.1	Front Matter	747
30.1.1	Preface	747
30.1.1.1	Conventions	747
30.1.1.2	Acronyms and Abbreviations	747
30.1.1.3	Glossary	748
30.2	Introduction	748
30.2.1	Overview	749
30.2.2	Features	749
30.3	Signal Description	749
30.4	Memory Map and Register Description	749
30.4.1	Memory Map	749
30.4.2	Register Descriptions	751
30.4.2.1	PIT Module Control Register (PITMCR)	751
30.4.2.2	Timer Load Value Register n (LDVALn)	752
30.4.2.3	Current Timer Value Register n (CVALn)	752
30.4.2.4	Timer Control Register n (TCTRLn)	753
30.4.2.5	Timer Flag Register n (TFLGn)	754
30.5	Functional Description	754

30.5.1	General	754
30.5.1.1	Timers	754
30.5.1.2	Debug Mode	755
30.5.2	Interrupts	755
30.6	Initialization and Application Information	756
30.6.1	Example Configuration	756

Chapter 31 Software Watchdog Timer (SWT)

31.1	Introduction	757
31.1.1	Overview	757
31.1.2	Features	757
31.1.3	Modes of operation	757
31.2	External signal description	757
31.3	Memory map and register definition	758
31.3.1	Memory map	758
31.3.2	Register descriptions	758
31.3.2.1	SWT Module Control Register (SWT_MCR)	758
31.3.2.2	SWT Interrupt Register (SWT_IR)	760
31.3.2.3	SWT Time-Out Register (SWT_TO)	760
31.3.2.4	SWT Window Register (SWT_WN)	761
31.3.2.5	SWT Service Register (SWT_SR)	761
31.3.2.6	SWT Counter Output Register (SWT_CO)	762
31.3.2.7	SWT Service Key Register (SWT_SK)	762
31.4	Functional description	763

Chapter 32 System Timer Module (STM)

32.1	Overview	765
32.2	Features	765
32.3	Modes of operation	765
32.4	External signal description	765
32.5	Memory map and registers description	765
32.5.1	Memory map	765
32.5.2	Registers description	766
32.5.2.1	STM Control Register (STM_CR)	766
32.5.2.2	STM Count Register (STM_CNT)	767
32.5.2.3	STM Channel Control Register (STM_CCR _{<i>n</i>})	768
32.5.2.4	STM Channel Interrupt Register (STM_CIR _{<i>n</i>})	768
32.5.2.5	STM Channel Compare Register (STM_CMP _{<i>n</i>})	769
32.6	Functional description	769

Chapter 33 Cyclic Redundancy Check (CRC)

33.1	Introduction	771
------	--------------------	-----

33.1.1	Glossary	771
33.2	Main features	771
33.2.1	Standard features	771
33.3	Block diagram	772
33.3.1	IPS bus interface	772
33.4	Functional description	773
33.5	Memory map and registers description	774
33.5.1	CRC Configuration Register (CRC_CFG)	775
33.5.2	CRC Input Register (CRC_INP)	776
33.5.3	CRC Current Status Register (CRC_CSTAT)	777
33.5.4	CRC Output Register (CRC_OUTP)	777
33.6	Use cases and limitations	778

Chapter 34 Boot Assist Module (BAM)

34.1	Overview	783
34.2	Features	783
34.3	Boot modes	783
34.4	Memory map	783
34.5	Functional description	784
34.5.1	Entering boot modes	784
34.5.2	Reset Configuration Half Word (RCHW)	784
34.5.3	Single chip boot mode	786
34.5.3.1	Boot and alternate boot	786
34.5.4	Boot through BAM	786
34.5.4.1	Executing BAM	786
34.5.4.2	BAM software flow	787
34.5.4.3	BAM resources	788
34.5.4.4	Download and execute the new code	789
34.5.4.5	Download 64-bit password and password check	789
34.5.4.6	Download start address, VLE bit and code size	791
34.5.4.7	Download data	792
34.5.4.8	Execute code	792
34.5.5	Boot from UART—autobaud disabled	792
34.5.5.1	Configuration	792
34.5.5.2	UART boot mode download protocol	793
34.5.6	Bootstrap with FlexCAN—autobaud disabled	793
34.5.6.1	Configuration	793
34.6	FlexCAN boot mode download protocol	794
34.6.1	Autobaud feature	794
34.6.1.1	Configuration	795
34.6.1.2	Boot from UART with autobaud enabled	795
34.6.1.2.1	Choosing the host baud rate	796
34.6.1.3	Boot from FlexCAN with autobaud enabled	799
34.6.1.3.1	Choosing the host baud rate	801

34.6.2 Interrupt	802
------------------------	-----

Chapter 35 Inter-Integrated Circuit Bus Controller Module (I2C)

35.1 Introduction	803
35.1.1 Overview	803
35.1.2 Features	803
35.1.3 Modes of Operation	804
35.1.4 Block Diagram	804
35.2 External Signal Description	804
35.2.1 Overview	804
35.2.2 Detailed Signal Descriptions	805
35.2.2.1 SCL	805
35.2.2.2 SDA	805
35.3 Memory Map/Register Definition	805
35.3.1 Overview	805
35.3.2 Module Memory Map	805
35.3.3 Register Descriptions	806
35.3.3.1 I ² C Address Register	806
35.3.3.2 I ² C Frequency Divider Register	806
35.3.3.3 I ² C Control Register	813
35.3.3.4 I ² C Status Register	814
35.3.3.5 I ² C Data I/O Register	815
35.3.3.6 I ² C Interrupt Config Register	816
35.4 Functional Description	816
35.4.1 General	816
35.4.2 I-Bus Protocol	816
35.4.2.1 START Signal	817
35.4.2.2 Slave Address Transmission	818
35.4.2.3 Data Transfer	818
35.4.2.4 STOP Signal	818
35.4.2.5 Repeated START Signal	819
35.4.2.6 Arbitration Procedure	819
35.4.2.7 Clock Synchronization	819
35.4.2.8 Handshaking	820
35.4.2.9 Clock Stretching	820
35.4.3 Interrupts	820
35.4.3.1 General	820
35.4.3.2 Interrupt Description	820
35.5 Initialization/Application Information	821
35.5.1 I ² C Programming Examples	821
35.5.1.1 Initialization Sequence	821
35.5.1.2 Generation of START	821
35.5.1.3 Post-Transfer Software Response	821
35.5.1.4 Generation of STOP	822

35.5.1.5	Generation of Repeated START	823
35.5.1.6	Slave Mode	823
35.5.1.7	Arbitration Lost	823

Chapter 36 Fast Ethernet Controller (FEC)

36.1	Overview	825
36.1.1	Features	827
36.2	Modes of Operation	828
36.2.1	Full- and Half-Duplex Operation	828
36.2.2	Interface Options	828
36.2.2.1	10-Mbps and 100-Mbps Media Independent Interface (MII)	828
36.2.3	Address Recognition Options	828
36.2.4	Internal Loopback	828
36.3	Memory Map and Register Definition	828
36.3.1	Top Level Module Memory Map	829
36.3.2	Detailed Memory Map (Control/Status Registers)	829
36.3.3	Message Information Block (MIB) Counters Memory Map	830
36.3.4	Register Descriptions	832
36.3.4.1	Ethernet Interrupt Event Register (EIR)	832
36.3.4.2	Ethernet Interrupt Mask Register (EIMR)	834
36.3.4.3	Receive Descriptor Active Register (RDAR)	835
36.3.4.4	Transmit Descriptor Active Register (TDAR)	836
36.3.4.5	Ethernet Control Register (ECR)	837
36.3.4.6	MII Management Frame Register (MMFR)	838
36.3.4.7	MII Speed Control Register (MSCR)	839
36.3.4.8	MIB Control Register (MIBC)	841
36.3.4.9	Receive Control Register (RCR)	841
36.3.4.10	Transmit Control Register (TCR)	843
36.3.4.11	Physical Address Low Register (PALR)	844
36.3.4.12	Physical Address Upper Register (PAUR)	845
36.3.4.13	Opcode/Pause Duration Register (OPDR)	845
36.3.4.14	Descriptor Individual Address Upper Register (IAUR)	846
36.3.4.15	Descriptor Individual Address Lower Register (IALR)	847
36.3.4.16	Descriptor Group Address Upper Register (GAUR)	847
36.3.4.17	Descriptor Group Address Lower Register (GALR)	848
36.3.4.18	Transmit FIFO Watermark Register (TFWR)	849
36.3.4.19	FIFO Receive Bound Register (FRBR)	849
36.3.4.20	FIFO Receive Start Register (FRSR)	850
36.3.4.21	Receive Buffer Descriptor Ring Start Register (ERDSR)	851
36.3.4.22	Transmit Buffer Descriptor Ring Start Register (ETDSR)	852
36.3.4.23	Maximum Receive Buffer Size Register (EMRBR)	852
36.4	Functional Description	853
36.4.1	Network Interface Options	853
36.4.2	FEC Frame Transmission	854

36.4.2.1	Transmit Inter-Packet Gap (IPG) Time	855
36.4.2.2	Collision Handling	855
36.4.2.3	Transmission Error Handling	855
36.4.2.3.1	Transmitter Underrun	855
36.4.2.3.2	Retransmission Attempts Limit Expired	856
36.4.2.3.3	Late Collision	856
36.4.2.3.4	Heartbeat	856
36.4.3	FEC Frame Reception	856
36.4.3.1	Receive Inter-Packet Gap (IPG) Time	857
36.4.3.2	Ethernet Address Recognition	857
36.4.3.2.1	Hash Algorithm	860
36.4.3.3	Reception Error Handling	863
36.4.3.3.1	Overrun	863
36.4.3.3.2	Non-Octet (Dribbling Bits)	863
36.4.3.3.3	CRC	863
36.4.3.3.4	Frame Length Violation	864
36.4.3.3.5	Truncation	864
36.4.4	Full-Duplex Flow Control	864
36.4.5	Internal and External Loopback	865
36.5	Initialization/Application Information	865
36.5.1	Initialization Sequence	865
36.5.1.1	Hardware Controlled Initialization	865
36.5.1.2	User Initialization (Prior to Asserting ECR[ETHER_EN])	866
36.5.1.3	Microcontroller Initialization	866
36.5.1.4	User Initialization (after asserting ECR[ETHER_EN])	867
36.5.2	Buffer Descriptors	867
36.5.2.1	Driver/DMA Operation with Buffer Descriptors	867
36.5.2.2	Ethernet Transmit Buffer Descriptor (TxBD)	868
36.5.2.2.1	Driver/DMA Operation with Transmit Buffer Descriptors	869
36.5.2.3	Ethernet Receive Buffer Descriptor (RxBD)	870
36.5.2.3.1	Driver/DMA Operation with Receive Buffer Descriptors	871

Chapter 37

IEEE 1588

37.1	Introduction	873
37.2	IEEE1588 Block Diagram	874
37.3	Time Stamp Unit (TSU) Key Features	874
37.4	IEEE1588 Real Time Clock (RTC) Key Features	875
37.5	IEEE1588 Implementation Assumptions	876
37.6	Modes of Operation	876
37.7	Memory Map/Register Definition	877
37.8	Time Stamp Unit Mode Registers	879
37.8.1	Time Stamp Unit Parsing Definitions Register 1 (PTP_TSPDR1)	879
37.8.2	Time Stamp Unit Parsing Definitions Register 2(PTP_TSPDR2)	880
37.8.3	Time Stamp Unit Parsing Definitions Register 3 (PTP_TSPDR3)	881

37.8.4	Time Stamp Unit Parsing Definitions Register 4 (PTP_TSPDR4)	882
37.8.5	Time Stamp Unit Parsing Definitions Register 5 (PTP_TSPDR5)	883
37.8.6	Time Stamp Unit Parsing Definitions Register 6 (PTP_TSPDR6)	884
37.8.7	Time Stamp Unit Parsing Definitions Register 7 (PTP_TSPDR7)	886
37.8.8	Time Stamp Unit Parsing Offset Values (PTP_TSPOV)	887
37.8.9	Time Stamp Unit Mode Register (PTP_TSMR)	889
37.8.10	Timer PTP Event Register (PTP_TMR_PEVENT)/ Timer PTP Mask Register (PTP_TMR_PEMASK)	890
37.8.11	Time Stamp Unit Receiver Time High (TMR_UC_RXTS_H)/Time Stamp Unit Receiver Time Low (TMR_UC_RXTS_L)/Time Stamp Unit Transmitter Time High (TMR_UC_TXTS_H)/Time Stamp Unit Transmitter Time Low (TMR_UC_TXTS_L)	893
37.9	IEEE1588 Timer Mode Registers	894
37.9.1	Timer Control Register (TMR_CTRL)	894
37.9.2	Timer Event Register (TMR_TEVENT)/Timer Event Mask Register (TMR_TEMASK)	896
37.9.3	Timer Counter Register (TMR_CNT_L/TMR_CNT_H)	898
37.9.4	Timer Addend Register (TMR_ADD)	899
37.9.5	Timer Accumulator Register (TMR_ACC)	900
37.9.6	Timer Prescale Register (TMR_PRSC)	901
37.9.7	Timer Offset Register (TMROFF_L/TMROFF_H)	902
37.9.8	Alarm Time Register (TMR_ALARM_L/TMR_ALARM_H)	903
37.9.9	Timer Fixed Interval Period Register (TMR_FIPERn)	903
37.9.10	FIPER Start Register (TMR_FSV_L/TMR_FSV_H)	905
37.9.11	External Trigger Time Stamp Register (TMR_ETTS_L/TMR_ETTS_H)	905
37.10	Time Stamp Unit (TSU)	906
37.10.1	PTP Event Interrupts	907
37.11	IEEE1588 Real Time Clock (RTC)	908
37.11.1	RTC Clock Sources	910
37.11.2	Prescale Output Clock and Pulse per Second Edge Alignment	910
37.12	PTP Frame Reception	910
37.12.1	Out-of-Band Mode	910
37.13	PTP Frame Transmission	911
37.14	Cycle Delay from Time Stamp Location	911
37.15	Initialization Sequence	911
37.15.1	TSU Mode Registers	911
37.15.2	RTC Mode Registers	912
37.15.3	Enable Sequence	912

Chapter 38 Nexus Debug Interface (NDI)

38.1	Overview	913
38.1.1	Features	913
38.2	Nexus Port Controller (NPC)	914
38.2.1	Features	914
38.2.2	Modes of Operation	914

38.2.2.1	Reset	915
38.2.2.2	Disabled-Port Mode	915
38.2.2.3	Full-Port Mode	915
38.3	External Signal Description	915
38.3.1	Overview	915
38.3.2	Detailed Signal Descriptions	916
38.3.2.1	EVTO_B - Event Out	916
38.3.2.2	MDO - Message Data Out	916
38.3.2.3	MSEO_B - Message Start/End Out	916
38.3.2.4	TCK - Test Clock Input	916
38.3.2.5	TDI - Test Data Input	916
38.3.2.6	TDO - Nexus Test Data Output	917
38.3.2.7	TMS - Test Mode Select	917
38.4	Register Definition	917
38.4.1	Register Descriptions	917
38.4.1.1	Bypass Register	917
38.4.1.2	Instruction Register	917
38.4.1.3	Nexus Device ID Register (DID)	918
38.4.1.4	Port Configuration Register (PCR)	919
38.5	Functional Description	921
38.5.1	NPC_HNDSHK module	921
38.5.2	NPC Reset Configuration	922
38.5.3	Auxiliary Output Port	922
38.5.3.1	Output Message Protocol	922
38.5.3.2	Output Messages	923
38.5.3.3	Rules of Message	924
38.5.4	IEEE 1149.1-2001 (JTAG) TAP	925
38.5.4.1	Enabling the NPC TAP controller	925
38.5.4.2	Retrieving device IDCODE	927
38.5.4.3	Loading NEXUS-ENABLE Instruction	927
38.5.4.4	Selecting a Nexus Client Register	928
38.5.5	Nexus JTAG Port Sharing	929
38.5.6	MCKO and ipg_sync_mcko	929
38.5.7	EVTO Sharing	929
38.5.8	Nexus Reset Control	929
38.6	Initialization/Application Information	929
38.6.1	Accessing NPC tool-mapped registers	929

Chapter 39 Register Protection (REG_PROT)

39.1	Introduction	931
39.1.1	Overview	931
39.1.2	Features	931
39.1.3	Modes of Operation	932
39.2	External Signal Description	932

39.3	Memory Map and Register Definition	932
39.3.1	Memory Map	933
39.3.2	Register Descriptions	934
39.3.2.1	Module Registers (MR0-6143)	934
39.3.2.2	Module Register and Set Soft Lock Bit (LMR0-6143)	934
39.3.2.3	Soft Lock Bit Register (SLBR0-1535)	934
39.3.2.4	Global Configuration Register (GCR)	935
39.4	Functional Description	936
39.4.1	General	936
39.4.2	Change Lock Settings	937
39.4.2.1	Change Lock Settings Directly Via Area #4	937
39.4.2.2	Enable Locking Via Mirror Module Space (Area #3)	938
39.4.2.3	Write Protection for Locking Bits	939
39.4.3	Access Errors	940
39.5	Initialization/Application Information	940
39.5.1	Reset	940
39.5.2	Writing C code using the register protection scheme	940
39.6	Registers under protection	942

Chapter 40 Temperature Sensor (TSENS)

40.1	Introduction	955
40.2	Features	955
40.3	Signals	955
40.4	Memory Map and Register Description	956
40.4.1	Memory Map	956
40.5	Modes of operation	956
40.6	Obtaining the device temperature using TSENS	956
40.6.1	TSENS calibration constants	956
40.6.2	Equations for converting TSENS voltage to device temperature	957

Chapter 41 JTAG Controller (JTAGC)

41.1	Introduction	959
41.1.1	Overview	959
41.1.2	Features	960
41.1.3	Modes of Operation	960
41.1.3.1	Reset	960
41.1.3.2	IEEE 1149.1-2001 Defined Test Modes	960
41.1.3.3	Bypass Mode	961
41.2	External Signal Description	961
41.2.1	Overview	961
41.2.2	Detailed Signal Descriptions	961
41.2.2.1	TCK - Test Clock Input	961
41.2.2.2	TDI - Test Data Input	961

41.2.2.3	TDO - Test Data Output	961
41.2.2.4	TMS - Test Mode Select	962
41.3	Register Definition	962
41.3.1	Register Descriptions	962
41.3.1.1	Instruction Register	962
41.3.1.2	Bypass Register	962
41.3.1.3	Device Identification Register	963
41.3.1.4	CENSOR_CTRL Register	963
41.3.1.5	Boundary Scan Register	964
41.4	Functional Description	964
41.4.1	JTAGC Reset Configuration	964
41.4.2	IEEE 1149.1-2001 (JTAG) Test Access Port	964
41.4.3	TAP Controller State Machine	965
41.4.3.1	Enabling the TAP Controller	967
41.4.3.2	Selecting an IEEE 1149.1-2001 Register	967
41.4.4	JTAGC Block Instructions	967
41.4.4.1	IDCODE Instruction	968
41.4.4.2	SAMPLE/PRELOAD Instruction	968
41.4.4.3	SAMPLE Instruction	968
41.4.4.4	EXTEST — External Test Instruction	968
41.4.4.5	ENABLE_CENSOR_CTRL Instruction	968
41.4.4.6	CLAMP Instruction	969
41.4.4.7	ACCESS_AUX_TAP_x Instructions	969
41.4.4.8	BYPASS Instruction	969
41.4.5	Boundary Scan	969
41.5	Initialization/Application Information	969

About This Book

This reference manual describes the MPC5604E processor for software and hardware developers. Information regarding bus timing, signal behavior, and AC, DC, and thermal characteristics are detailed in the device data sheet (*MPC5604E Microcontroller Data Sheet*).

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

To locate any published errata or updates for this document, refer to the world-wide web at <http://www.freescale.com/powerpc>.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5604E processor. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the PowerPC[®] architecture.

Chapter Organization and Device-Specific Information

This document includes chapters that describe:

- The device as a whole
- The functionality of the individual modules on the device

In the latter, any device-specific information is presented in the section “Information Specific to This Device” at the beginning of the chapter, where the chapter may describe a superset of features.

Suggested Reading

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

General Information

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

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

PowerArchitecture Documentation

Power Architecture documentation is available from the sources listed on the back cover of this manual, as well as our web site, <http://www.freescale.com/powerpc>.

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

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

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
reserved	When a bit or address is reserved, it should not be written. If read, its value cannot be not guaranteed. Reading or writing to reserved bits or addresses may cause unexpected results.
MNEMONICS	In text, instruction mnemonics are shown in uppercase.
mnemonics	In code and tables, instruction mnemonics are shown in lowercase.
<i>italics</i>	Italics indicate variable command parameters. Book titles in text are set in italics.
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
REG[FIELD]	Abbreviations for registers are shown in uppercase. Specific bits, fields, or ranges appear in brackets. For example, RAMBAR[BA] identifies the base address field in the RAM base address register.
nibble	A 4-bit data unit
byte	An 8-bit data unit
halfword	A 16-bit data unit ¹

word	A 32-bit data unit
doubleword	A 64-bit data unit
x	In some contexts, such as signal encodings, x (without italics) indicates a “don’t care” condition.
<i>x</i>	With italics, used to express an undefined alphanumeric value (e.g., a variable in an equation); or a variable alphabetic character in a bit, register, or module name (e.g., DSPI_ <i>x</i> could refer to DSPI_A or DSPI_B).
<i>n</i>	Used to express an undefined numerical value; or a variable numeric character in a bit, register, or module name (e.g., EIF <i>n</i> could refer to EIF1 or EIF0).
~	NOT logical operator
&	AND logical operator
	OR logical operator
	Field concatenation operator
<u>OVERBAR</u>	An overbar indicates that a signal is active-low.

Register Figure Conventions

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

w1c	Write 1 to clear the bit to 0.
—	Undefined at reset or “not applicable.”
U	Bit value is uninitialized upon reset.
u	Bit value is unchanged upon reset.
[<i>signal_name</i>]	Reset value is determined by the polarity of the indicated signal.

The following register fields are used:

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

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

R	FIELDNAME	Indicates a read/write bit.
W		

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

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

R	
W	FIELDNAME

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

R	FIELDNAME
W	w1c

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

R	0
W	FIELDNAME

Indicates a self-clearing bit.

Acronyms and Abbreviations

Table i lists acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADC	Analog-to-digital conversion
ALU	Arithmetic logic unit
BDM	Background debug mode
BIST	Built-in self test
BSDL	Boundary-scan description language
CODEC	Code/decode
DAC	Digital-to-analog conversion
DMA	Direct memory access
DSP	Digital signal processing
EA	Effective address
FIFO	First-in, first-out
GPIO	General-purpose I/O
IEEE	Institute for Electrical and Electronics Engineers
IFP	Instruction fetch pipeline
IPL	Interrupt priority level
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LIFO	Last-in, first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiply accumulate unit, also Media access controller
MSB	Most-significant byte

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
msb	Most-significant bit
Mux	Multiplex
NC	No connection
NOP	No operation
OEP	Operand execution pipeline
PC	Program counter
PLIC	Physical layer interface controller
PLL	Phase-locked loop
PIN	Referring to an external pin or ball (i.e. external signal)
POR	Power-on reset
RISC	Reduced instruction set computing
Rx	Receive
SOF	Start of frame
STAC	Shared Time and Counter
TAP	Test access port
TTL	Transistor transistor logic
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
USB	Universal serial bus

Terminology Conventions

Table ii shows terminology conventions used throughout this document.

Table ii. Notational Conventions

Instruction	Operand Syntax
Opcode Wildcard	
cc	Logical condition (example: NE for not equal)
Register Specifications	
An	Any address register n (example: A3 is address register 3)
Ay,Ax	Source and destination address registers, respectively
Dn	Any data register n (example: D5 is data register 5)
Dy,Dx	Source and destination data registers, respectively
Rc	Any control register (example VBR is the vector base register)

Table ii. Notational Conventions (continued)

Instruction	Operand Syntax
Rm	MAC registers (ACC, MAC, MASK)
Rn	Any address or data register
Rw	Destination register w (used for MAC instructions only)
Ry,Rx	Any source and destination registers, respectively
Xi	Index register i (can be an address or data register: Ai, Di)
Miscellaneous Operands	
#<data>	Immediate data following the 16-bit operation word of the instruction
<ea>	Effective address
<ea>y,<ea>x	Source and destination effective addresses, respectively
<label>	Assembly language program label
<list>	List of registers for MOVEM instruction (example: D3–D0)
<shift>	Shift operation: shift left (<<), shift right (>>)
<size>	Operand data size: byte (B), word (W), longword (L)
bc	Instruction and data caches
dc	Data cache
ic	Instruction cache
# <vector>	Identifies the 4-bit vector number for trap instructions
<>	identifies an indirect data address referencing memory
<xxx>	identifies an absolute address referencing memory
dn	Signal displacement value, <i>n</i> bits wide (example: d16 is a 16-bit displacement)
SF	Scale factor (x1, x2, x4 for indexed addressing mode, <<1n>> for MAC operations)
Operations	
+	Arithmetic addition or postincrement indicator
–	Arithmetic subtraction or predecrement indicator
x	Arithmetic multiplication

Table ii. Notational Conventions (continued)

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

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 1

Overview

1.1 Chipset overview

The MPC5604E microcontroller is a gateway system designed to move data from different sources via Ethernet to a receiving system and vice versa. The supported data sources and sinks are:

- Video data (with 8/10/12 bits per data word)
- Audio data (6× stereo channels)
- RADAR data (2 × 12 bit with <math><1\mu\text{s}</math> per sample, digitized externally and read in via SPI)
- Other serial communication interfaces including CAN, LIN, and SPI

The Ethernet module has a bandwidth of 10/100 Mbits/sec and supports precision time stamps (IEEE1588). Unshielded twisted pair cables are used to transfer data (via Ethernet) in the car, resulting in a significant reduction of wiring costs by providing inexpensive high bandwidth data links.

The core selected for the device is the Harvard bus interface version of the e200z0 to cover the low-end chassis application space.

The e200 processor family is a set of CPU cores that implement low-cost versions of the Power Architecture Book E architecture. The e200 processors are designed for deeply embedded control applications that require low cost solutions rather than maximum performance. The e200z0 processor integrates an integer execution unit, branch control unit, instruction fetch and load/store units, and a multi-ported register file capable to sustaining three read and two write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by branch unit to allow single-cycle branches in some cases. The e200z0 core is a single-issue, 32-bit Power Architecture Book E VLE only design with 32-bit general purpose registers (GPRs). All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). Instead of the base Power Architecture Book E instruction set support, the e200z0 core only implements the VLE (variable length encoding) APU, providing improved code density.

The MPC5604E has a single level of memory hierarchy consisting of 96 KB on-chip SRAM and 578 KB (512 KB code + 64 KB data) of on-chip flash memory. Both the SRAM and the flash memory can hold instructions and data.

Multimedia support is provided by a video encoder module and a 6× stereo audio (SAI) module.

The timer functions of MPC5604E are performed by the eTimer Modular Timer System and Peripheral Interrupt Timer (PIT) modules. The eTimer module implements enhanced timer features (six channels) including dedicated motor control quadrature decode functionality and DMA support. The PIT module includes four general purpose interrupt timers (32-bit counters) with DMA support for each channel.

Off-chip communication is performed by a suite of serial protocols including CANs, ethernet, enhanced SPIs (DSPI), and SCIs (LINFlex).

The System Integration Unit Lite (SIUL) performs several chip-wide configuration functions. Pad configuration and general-purpose input/output (GPIO) are controlled from the SIUL. External interrupts are also found in the SIUL.

As the MPC5604E is built on a wider legacy of Power Architecture-based devices, when applicable and possible, reusing or enhancement of existing IP, design and concepts is adopted.

1.2 Target applications

This device is a gateway system to move data from different sources via Ethernet to a receiving system and vice versa. The supported data sources/sinks combined with the Ethernet are:

- Video data (with 8/10/12 bits per data word)
- Audio data (6x stereo channels)
- RADAR data (2x 12 bit with <1us per sample, digitized externally and read in via SPI)
- Other Serial communication interfaces like CAN, LIN, and SPI

The Ethernet has a bandwidth of 10/100 Mbits/sec supporting precision time stamps (IEEE1588). Unshielded twisted pair cables are then used to transfer information (via Ethernet) in the car. Thus, a significant reduction of wiring costs in the car can be achieved by providing high bandwidth data links.

The Ethernet AVB is an upcoming high-bandwidth communication standard in the automotive area competing with established protocols like LVDS, MOST, and FlexRay (to a sudden extend for some chassis applications).

1.3 Features

The table provides a summary of the features of the MPC5604E.

Table 1. Device summary

Feature	MPC5604E	
	100-pin LQFP ¹	64-pin LQFP
CPU	e200z0h, 64 MHz, VLE only, no SPE	
Flash with ECC	CFlash: 512 KB (LC) DFlash: 64 KB (LC, area optimized)	
RAM with ECC	96 KB	
DMA	16 channels	
PIT	yes	
SWT	yes	
FCU	yes	
Ethernet	100 Mbits MII	100 Mbits MII-Lite
Video Encoder	8bpp/12bpp	

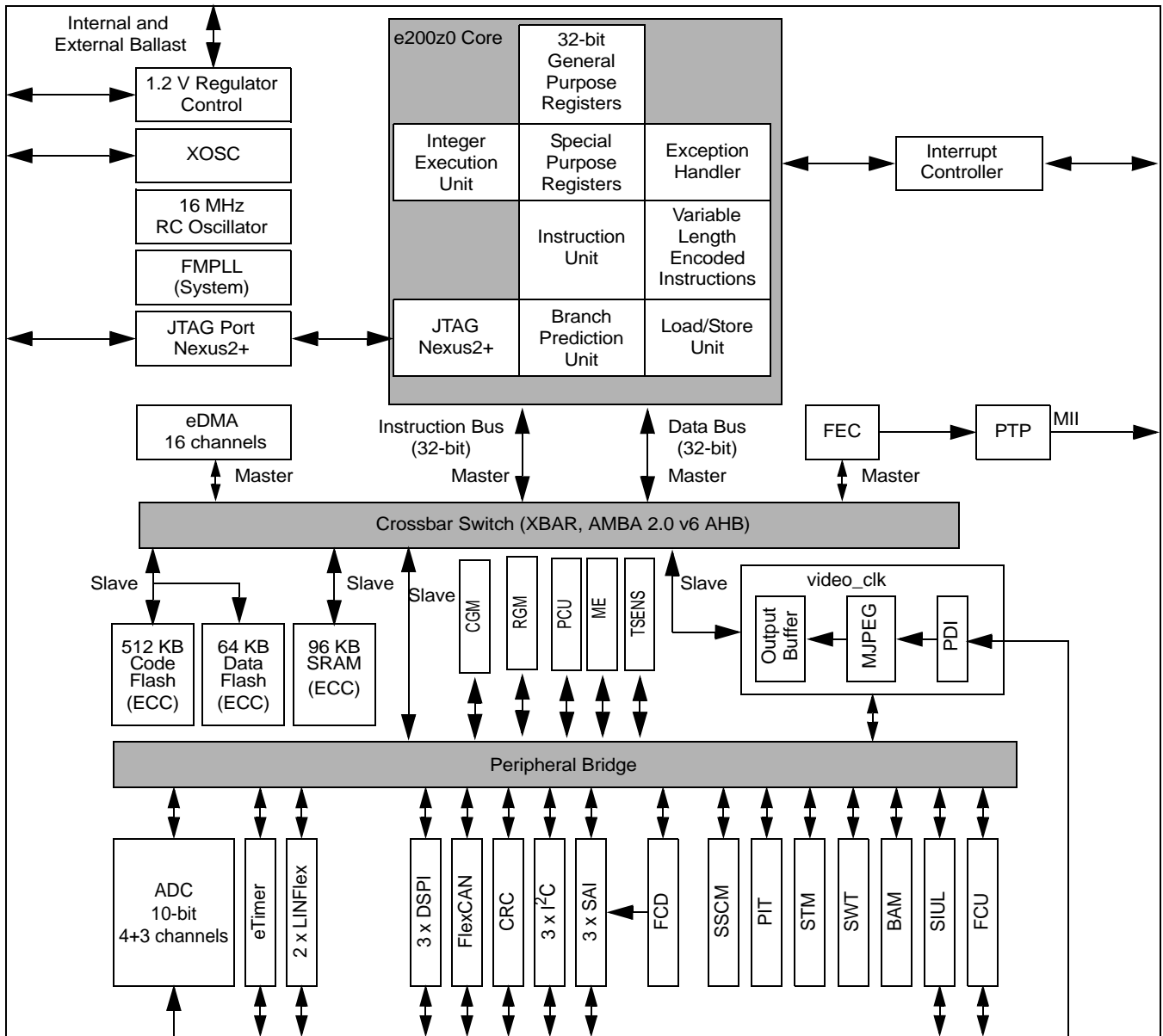
Table 1. Device summary (continued)

Feature	MPC5604E	
	100-pin LQFP ¹	64-pin LQFP
Audio Interface	6x Stereo (4x synchronous + 2x synchronous/asynchronous)	
ADC (10-bit)	1× 4 channels + V _{DD_IO} + V _{DDCore} + TSens	
Timer I/O (eTimer)	1×6 channels	
SCI (LINFlex)	2×	
SPI (DSPI)	DSPI_0: 2 chip selects DSPI_1: 2 chip selects DSPI_2: 4 chip selects	
CAN (FlexCAN)	1×	
IIC	2×	
Supply	3.3 V IO 1.2V Core with dedicated ballast source pin in two modes: <ul style="list-style-type: none"> • internal ballast or • external supply (using power on reset pin) 	
Phase Lock Loop (PLL)	1× FMPLL	
Internal RC Oscillator	16 MHz	
External crystal Oscillator	4 MHz - 40 MHz	
CRC	yes	
Debug	JTAG, Nexus2+	JTAG
Ambient Temperature	-40 to 125 °C	

¹ The 100-pin package is not a production package. It is used for software development only.

1.4 Block diagram

Figure 1 shows a top-level block diagram of the MPC5604E microcontroller.



- | | | | |
|---------|--|-------|--|
| ADC | Analog-to-Digital Converter | CGM | Clock Generation Module |
| BAM | Boot Assist Module | PCU | Power Control Unit |
| CRC | Cylic Redundancy Check | RGM | Reset Generation Module |
| DSPi | Deserial Serial Peripheral Interface | TSENS | Temperature sensor |
| eDMA | Enhanced Direct Memory Access | MJPEG | 12-bit Motion JPEG Encoder |
| eTimer | Enhanced Timer | PDI | Parallel Data Interface (image sensor) |
| FCD | Fractional Clock Divider | PIT | Periodic Interrupt Timer |
| FCU | Fault Collection Unit | PTP | IEEE 1588 Precision Time Stamps |
| FEC | Fast Ethernet Controller | SIUL | System Integration Unit |
| FlexCAN | Flexible Controller Area Network | SRAM | Static Random-Access Memory |
| FMPLL | Frequency-Modulated Phase-Locked Loop | SSCM | System Status and Configuration Module |
| I2C | Inter-Integrated Circuit serial interface | STM | System Timer Module |
| SAI | Serial Audio Interface 6xStereo | SWT | Software Watchdog Timer |
| LINFlex | Serial Communication Interface (LIN support) | | |
| ME | Mode Entry Module | | |

Figure 1. MPC5604E block diagram

1.5 Application examples

The following sections contain examples of applications for the MPC5604E microcontroller.

1.5.1 CMOS vision sensor gateway

The active safety and advanced driver assistance systems (ADAS) support Panorama View Park-Assist providing a high quality view of the vehicle's surroundings (typically a bird's eye view).

For this, up to 5 CMOS cameras with wide-angle lenses attached to the car. A typical installation has one camera at each corner of the front bumper, one in each side mirror and one in the rear. The front-viewing sensors cannot be combined with the front-viewing camera sensor used for active safety applications due to completely different optical requirements. All sensors are connected to a central fusion Electronic Control Unit (ECU) that performs enhancement and image generation.

First, the fusion unit corrects the wide-angle distortion in each image, if not done optically. Alternatively, there is an inexpensive optical solution (2nd inverting lens) on the market.

The next step is the stitching of the images—similar to the feature found on many of today's digital cameras. There is a broad range of algorithm complexity depending on the required quality. In principle, similarities in adjacent images need to be identified, e.g., by running matching filters. After identifying how the images fit geometrically together, there is some post-processing necessary for a smooth appearance within the overlapping areas.

Finally, the stitched images are rendered on a 3D grid model representing the chosen perspective to generate the final image.

The interconnect between the remote cameras and central fusion unit is done in a point-to-point manner with a switch located in the central ECU. The switch combines the Ethernet AVB streams and sends them, e.g., via GigE, to the central processing unit. Future systems with more ADAS nodes (e.g., cameras and RADARs in the bumper) might have two dedicated ADAS switches.

Figure 2 illustrates a multi-camera system based on the MPC5604E.

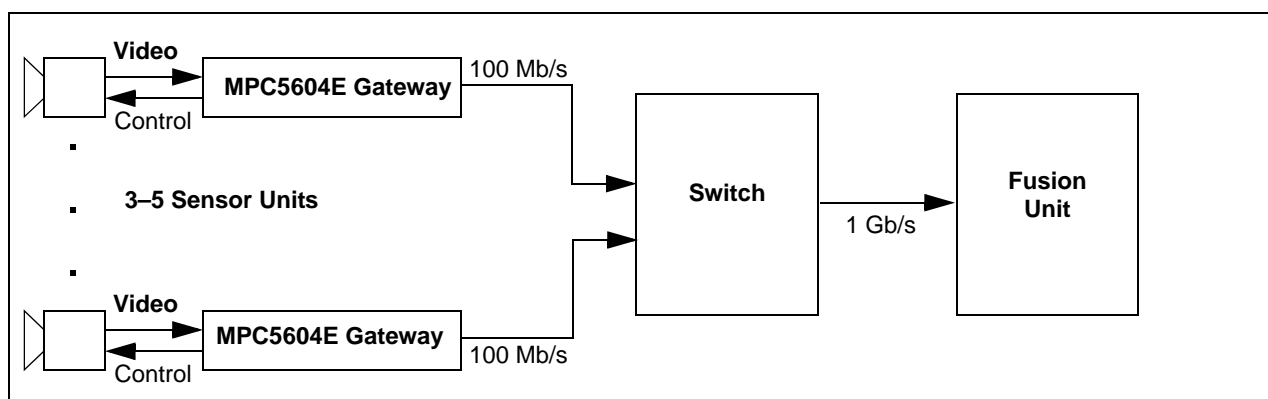


Figure 2. Multi-camera system level diagram

Each camera in Figure 2 is connected to one MPC5604E gateway via a parallel digital interface as shown in Figure 3. The raw data is buffered and the color component is vertically sub-sampled from YUV4:2:2 to YUV4:2:0. A low latency video encoder compresses the image data by a factor of 1:5/1:10 or higher

into a bit stream. This compression is not lossless, thus, the quality of the image is degraded with higher compression ratios. The video bitstream is then buffered in the MPC5604E (dedicated video bit stream buffer) and transmitted via the Ethernet AVB link.

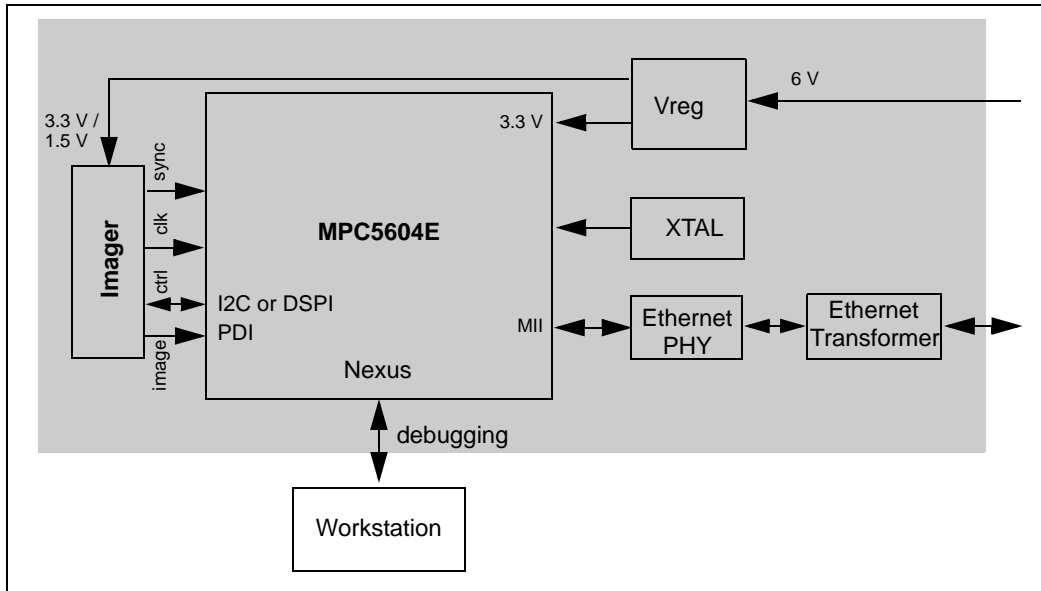
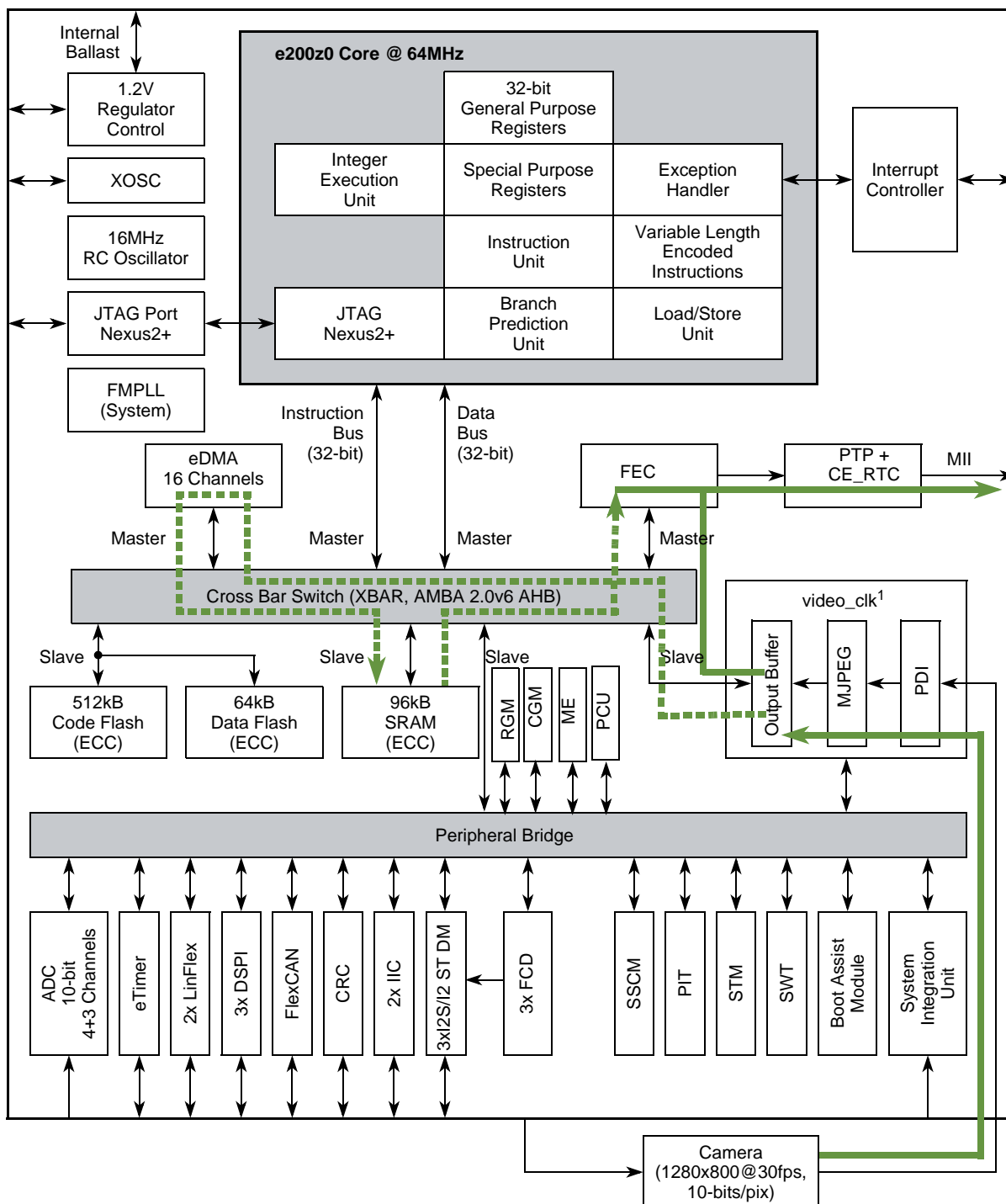


Figure 3. MPC5604E interfacing for CMOS sensor gateway

Figure 4 illustrates the processing flow of video data within the MPC5604E.



1. video_clk frequency can be 120/128 MHz depending on the system_clk (60/64 MHz).

Video Encoding:

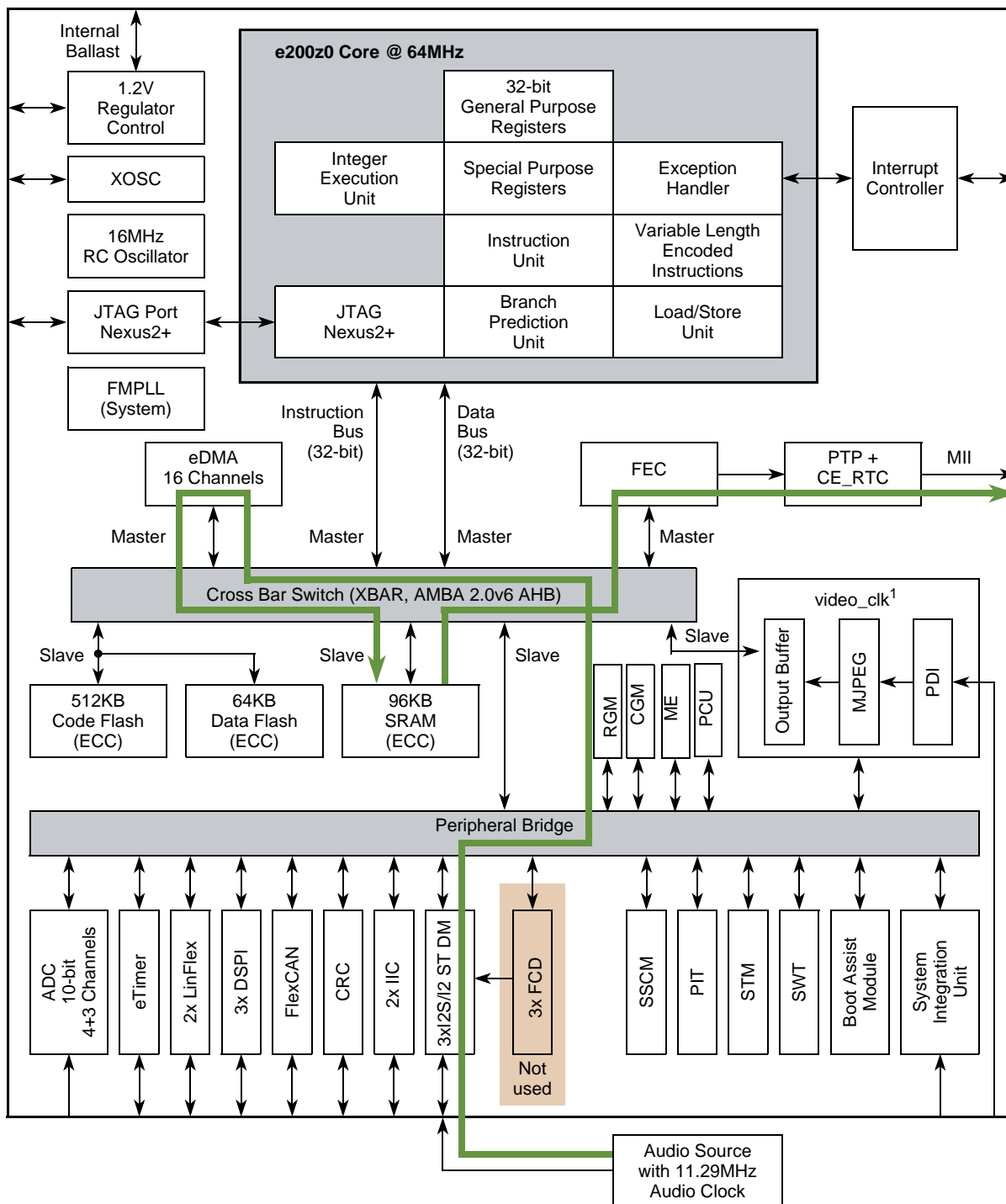
Video data is captured by the camera, and streamed via the PDI to the Video Encoder (MJPEG). The MJPEG offers the encoded data via an output buffer memory to the FEC (Ethernet).

Besides the video data, also histograms (for exposure control) are streamed to the PDI. The PDI separates these data that are moved via DMA to the SRAM. The CPU processes the information and set up the exposure/white balance via the IIC in the camera.

Figure 4. MPC5604E video data path

1.6 Audio source gateway

The MPC5604E can be effectively used as an audio source gateway. [Figure 5](#) shows the data flow for this application. Six stereo input audio channels at 44.1 KHz or 48 KHz are provided via I2S by an external audio source (radio, CD/DVD player, etc.). The external device provides the clock for its data (master).



1. video_clk frequency can be 120/128 MHz depending on the system_clk (60/64 MHz).

Audio In:

Data is sampled based on the input signals SAI_BCLK (11.29MHz) and SAI_SYNC. The I2S module (SAI) buffers the input data (6 channels) and signals the availability of data to the DMA module (in 64MHz domain). The DMA module moves the data to the SRAM. From here the FEC can move the data via the MII interface to a receiver.

Figure 5. Audio to ethernet data path

1.7 Critical performance parameters

MPC5604E is running under the following critical performance corner points:

- Maximum CPU frequency: 64 MHz
- Junction temperature range: $-40\text{ }^{\circ}\text{C}$ to $150\text{ }^{\circ}\text{C}$ ¹
- Nominal power dissipation: Less than 1.5 W
- Supply voltages:
 - $V_{DD_HV_IO} = 3.3\text{V}$
 - $V_{DD_HV_ADC} = 3.3\text{V}$
 - $V_{DD_LV_CORE} = 1.2\text{ V}$ (with internal ballast or external supply)

1.8 Chip-level features

On-chip modules available within the family include the following features:

- 32-bit Power Architecture[®] embedded CPU (e200z0h) with single issue and Harvard architecture
- Memory
 - 512 KB on-chip Code Flash with ECC and erase/program controller
 - additional 64 (4×16) KB on-chip Data Flash with ECC for EEPROM emulation
 - 96 KB on-chip SRAM with ECC
- Fail-safe protection
 - Programmable watchdog timer
 - Non-maskable interrupt
 - Fault collection unit
- Nexus 2+ interface
- Interrupts and events
 - 16-channel eDMA controller
 - 16 priority level controller
 - Up to 25 external interrupts
 - PIT implements four 32-bit timers
 - 120 interrupts are routed via INTC
- General purpose I/Os
 - Individually programmable as input, output or special function
 - 39 on LQFP64
 - 71 on LQFP100²
- 1 general purpose eTimer unit
 - 6 timers each with up/down capabilities

1. Ambient temperature is $125\text{ }^{\circ}\text{C}$, for the video use case with internal core voltage supply the ambient temperature is $105\text{ }^{\circ}\text{C}$.

2. The 100-pin package is not a production package. It is used for software development only.

- 16-bit resolution, cascadeable counters
- Quadrature decode with rotation direction flag
- Double buffer input capture and output compare
- Communications interfaces
 - 2 LINFlex channels (1 × Master/Slave, 1 × Master Only)
 - 3 DSPI channels with automatic chip select generation (up to 2/2/4 chip selects)
 - 1 FlexCAN interface (2.0B Active) with 32 message buffers
- One 10-bit analog-to-digital converter (ADC)
 - 7 input channels: 4 channels routed to the pins, 3 internal connections (Temperature sensor, Core voltage, IO voltage)
 - Conversion time < 1 μs including sampling time at full precision
 - 4 analog watchdogs with interrupt capability
- On-chip CAN/UART bootstrap loader with Boot Assist Module (BAM)
- 100 MBit Fast Ethernet Controller (FEC)
 - Supports precision timestamps
 - MII on 100-pin¹ LQFP package
 - MII-lite on 64-pin LQFP package
- Video encoder
- 6x stereo audio interface
- I²C controller module
- CRC module

Module features

1.8.1 High performance e200z0 core CPU

The e200z0 Power Architecture core provides the following features:

- High performance e200z0 core processor for managing peripherals and interrupts
- Single issue 4-stage pipeline in-order execution 32-bit Power Architecture CPU
- Harvard architecture
- Variable length encoding (VLE), allowing mixed 16-bit and 32-bit instructions
- Results in smaller code size footprint
- Minimizes impact on performance
- Branch processing acceleration using lookahead instruction buffer
- Load/store unit
- 1-cycle load latency
- Misaligned access support

1. The 100-pin package is not a production package. It is used for software development only.

- No load-to-use pipeline bubbles
- Thirty-two 32-bit general purpose registers (GPRs)
- Separate instruction bus and load/store bus Harvard architecture
- Hardware vectored interrupt support
- Reservation instructions for implementing read-modify-write constructs
- Long cycle time instructions, except for guarded loads, do not increase interrupt latency
- Extensive system development support through Nexus debug port
- Non Maskable Interrupt support

1.8.2 Crossbar switch (XBAR)

The XBAR multi-port crossbar switch supports simultaneous connections between four master ports and four slave ports. The crossbar supports a 32-bit address bus width and a 32-bit data bus width.

The crossbar allows for concurrent transactions to occur from any master port to any slave port. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will be stalled until the higher priority master completes its transactions. The default priority scheme is fixed priority based on the master ID. Besides this, the software can select a round robin arbitration.

The crossbar provides the following features:

- Four master ports
 - e200z0 core complex Instruction port
 - e200z0 core complex Load/Store Data port
 - eDMA
 - Ethernet
- Four slave ports
 - Flash memory (code flash and data flash) controller
 - SRAM controller
 - Video encoder output buffer
 - Peripheral bridge
- 32-bit internal address, 32-bit internal data paths
- Fixed Priority Arbitration based on port master
- Temporary dynamic priority elevation of masters

1.8.3 System clocks and clock generation

The following list summarizes the system clock and clock generation on the MPC5604E:

- Lock detect circuitry continuously monitors lock status
- Loss of clock (LOC) detection for PLL outputs
- Programmable output clock divider ($\div 1$, $\div 2$, $\div 4$, $\div 8$)

- Fractional clock divider clock for close loop controlled clocks
 - Provides audio clock in medium quality mode (approximately 11.29 MHz)
 - Provides camera input clock (25–30 MHz)
- On-chip oscillator with automatic level control
- Internal 16 MHz RC oscillator for rapid start-up and safe mode
 - Supports frequency trimming by user application
- Ethernet TX clock as input for the PLL (via OSC input pin)
- Up to 64 MHz for system clock; up to 128 MHz for video encoder clock

1.8.4 Frequency Modulated Phase Lock Loop (FMPLL)

The FMPLL allows the user to generate high speed system clocks from a 4 MHz to 40 MHz input clock. Further, the FMPLL supports programmable frequency modulation of the system clock. The PLL multiplication factor, output clock divider ratio are all software configurable.

The PLL has the following major features:

- Input clock frequency from 4 MHz to 40 MHz
- Voltage controlled oscillator (VCO) range from 256 MHz to 512 MHz
- Reduced frequency divider (RFD) for reduced frequency operation without forcing the PLL to re-lock
- Frequency modulated PLL
- Modulation enabled/disabled through software
- Triangle wave modulation
- Programmable modulation depth ($\pm 0.25\%$ to $\pm 2\%$ deviation from center frequency)
- Programmable modulation frequency dependent on reference frequency
- Self-locked mode (SCM) operation

1.8.5 Main oscillator

The main oscillator provides these features:

- Input frequency range 4 MHz to 40 MHz
- Crystal input mode or Oscillator input mode
- PLL reference

1.8.6 Internal RC oscillator

This device has an RC ladder phase-shift oscillator. The architecture uses constant current charging of a capacitor. The voltage at the capacitor is compared by the stable bandgap reference voltage.

The RC Oscillator provides these features:

- Nominal frequency 16 MHz
- $\pm 5\%$ variation over voltage and temperature after process trim

- Clock output of the RC oscillator serves as system clock source in case loss of lock or loss of clock is detected by the PLL
- RC oscillator is used as the default system clock during startup

1.8.7 Voltage regulator (VREG)

The on-chip voltage regulator module provides the following features:

- Available in two modes
 - Using internal PMOS ballast transistor to regulate external 3.3 V down to 1.2 V for the core logic
 - Disabled for using external supply for core logic
- Low voltage detection on the internal 1.2 V and I/O voltage 3.3 V

1.8.8 System Integration Unit (SIU-Lite)

The MPC5604E SIU-Lite controls MCU pad configuration, external interrupt, general purpose I/O (GPIO), and internal peripheral multiplexing.

The pad configuration block controls the static electrical characteristics of I/O pins. The GPIO block provides uniform and discrete input/output control of the I/O pins of the MCU.

The SIU provides the following features:

- Centralized general purpose input output (GPIO) control
 - 71 GPIO pads (bonding to pins is package dependent)
- As many as four internal output functions can be multiplexed onto one pin
- All GPIO pins can be independently configured to support pull-up, pull down, or no pull
- Reading and writing to GPIO supported both as individual pins and 16-bit wide ports
- All peripheral pins can be alternatively configured as both general purpose input or output pins
- Direct readback of the pin value is supported on all pins through the SIU supporting 4 external interrupts based on general purpose input pins
- Supports 4 external interrupts based on general purpose input pins (8 pads per interrupt)
- Configurable digital input filter that can be applied to general purpose input pins with interrupt functions for noise elimination

1.8.9 Boot Assist Module (BAM)

The BAM is a block of read-only one-time programmed memory and is identical for all MPC56XX devices that are based on the e200z0h core. The BAM program is executed every time the device is powered-on if the alternate boot mode has been selected by the user.

The BAM provides the following features:

- Boot from Internal Code Flash
 - Selected as default (using internal pull down on FAB pin).

- Censorship mode to protect the content of the flash memory.
- Alternate serial boot-loading via FlexCAN, LINFlex
 - BAM accepts a password via the used serial communication channel to grant the legitimate user access to the non-volatile memory.

1.8.10 Junction temperature sensor

The MPC5604E has a junction temperature sensor to enable measurement of the temperature of the silicon via the ADC.

The junction temperature sensor has these key parameters:

- Nominal temperature range from $-40\text{ }^{\circ}\text{C}$ to $150\text{ }^{\circ}\text{C}$
- Calibrated sensor accuracy:
 - $\pm 10\text{ }^{\circ}\text{C}$, -40 to $25\text{ }^{\circ}\text{C}$ ambient
 - $\pm 7\text{ }^{\circ}\text{C}$, 25 to $125\text{ }^{\circ}\text{C}$ ambient

1.8.11 JTAG controller (JTAGC)

The JTAG controller (JTAGC) block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. All data input to and output from the JTAGC block is communicated in serial format. The JTAGC block is compliant with the IEEE standard.

The JTAG controller provides the following features:

- IEEE Test Access Port (TAP) interface with four pins (TDI, TMS, TCK, TDO)
- Selectable modes of operation include JTAGC/debug or normal system operation.
- A 5-bit instruction register that supports the following IEEE 1149.1-2001 defined instructions:
 - BYPASS
 - IDCODE
 - EXTEST
 - SAMPLE
 - SAMPLE/PRELOAD
- A 5-bit instruction register that supports the additional following public instructions:
 - ACCESS_AUX_TAP_NPC
 - ACCESS_AUX_TAP_ONCE
- Three test data registers: a bypass register, a boundary scan register, and a device identification register.
- A TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

1.8.12 Nexus Debug Interface (NDI)

The NDI (Nexus Debug Interface) block provides real-time development support capabilities for the device Power Architecture based MCU in compliance with the IEEEISTO 5001-2003 standard. This development support is supplied for MCUs without requiring external address and data pins for internal visibility. The NDI block is an integration of several individual Nexus blocks that are selected to provide the development support interface for this device. The NDI block interfaces to the host processor and internal busses to provide development support as per the IEEE-ISTO 5001-2003 Class 2+ standard. The development support provided includes access to the MCUs internal memory map and access to the processors internal registers during run time.

The Nexus Interface provides the following features:

- Configured via the IEEE 1149.1
- All Nexus port pins operate at V_{DDIO} (no dedicated power supply)
- Nexus 2+ features supported
- Static debug
- Watchpoint messaging
- Ownership trace messaging
- Program trace messaging
- Real time read/write of any internally memory mapped resources through JTAG pins
- Overrun control, which selects whether to stall before Nexus overruns or keep executing and allow overwrite of information
- Watchpoint triggering for program tracing
- Auxiliary Output Port
- Four MDO (Message Data Out) pins in full port mode
- MCKO (Message Clock Out) pin
- Two MSEO (Message Start/End Out) pins
- EVTO (Event Out) pin
- Auxiliary Input Port
- EVTI (Event In) pin

1.8.13 DMA controller

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data movements via 16 programmable channels, with minimal intervention from the host processor. The hardware micro architecture includes a DMA engine which performs source and destination address calculations, and the actual data movement operations, along with an SRAM-based memory containing the transfer control descriptors (TCD) for the channels. This implementation is utilized to minimize the overall block size.

The eDMA module provides the following features:

- 16 channels support independent 8-, 16-, or 32-bit single value or block transfers

- Supports variable sized queues and circular queues
- Source and destination address registers are independently configured to post-increment or remain constant
- Each transfer is initiated by a peripheral, CPU, or eDMA channel request
- Each eDMA channel can optionally send an interrupt request to the CPU on completion of a single value or block transfer
- DMA transfers possible between system memories, DSPIs, ADC, eTimer, audio interface, and video bit stream output buffer
- Programmable DMA channel mux allows assignment of any DMA source to any available DMA channel with as many as 30 potential request sources.
- eDMA abort operation through software

1.8.14 DMA channel multiplexer (DMA_MUX)

- 32 independently selectable DMA channel routers
- Each channel router is assigned to one of the following sources:
 - One of the peripheral DMA sources
 - The always enabled source

1.8.15 Software Watchdog Timer (SWT)

The SWT on the MPC5604E is configured as the SWT found on MPC5604P devices. This includes, e.g., the reset values for the timer clock selection.

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Timer running on IRC clock for increased functional safety
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

1.8.16 System Timer Module (STM)

The STM module implements these features:

- 32-bit up counter with 8-bit pre-scaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

1.8.17 Periodic Interrupt Timers (PIT)

The PIT module implements these features:

- As many as four general purpose interrupt timers
- 32-bit counter resolution
- Clocked by system clock frequency
- Each channel can be used as trigger for a DMA request

1.8.18 FlexCAN module

The MPC5604E MCU contains one controller area network (FlexCAN) module. This module is a communication controller implementing the CAN protocol according to Bosch Specification version 2.0B. The CAN protocol was designed to be used primarily as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FlexCAN module contains 32 message buffers.

The FlexCAN module provides the following features:

- Supports the full implementation of the CAN Specification Version 2.0, Part B
 - Standard data and remote frames (up to 109 bits long)
 - Extended data and remote frames (up to 127 bits long)
 - 0 to 8 bytes data length
 - Programmable bit rate up to 1 Mbit/s
 - Content-related addressing
- 32 message buffers of 0 to 8 bytes data length
- Each message buffer configurable as RX or TX, all supporting standard and extended messages
- Listen-only mode capability
- Individual mask registers for each message buffer
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- Timestamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable loop-back mode supporting self-test operation
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- High immunity to EMI
- Short latency time due to an arbitration scheme for high-priority messages
- Wake-up when activity on the RX pin
 - Requires an external glitch filter at the pad (2750 ns of 0-input)
 - Wake-up via CAN interrupt
- Transmit features
 - Supports configuration of multiple mailboxes to form message queues of scalable depth

- Arbitration scheme according to message ID or message buffer number
- Internal arbitration to guarantee no inner or outer priority inversion
- Receive features
 - Individual programmable filters for each mailbox
 - Eight mailboxes configurable as a six-entry receive FIFO
 - Eight programmable acceptance filters for receive FIFO
- Programmable clock source
 - System clock
 - Direct oscillator clock to avoid PLL jitter

1.8.19 Deserial Serial Peripheral Interface (DSPI)

The deserial serial peripheral interface (DSPI) module provides a synchronous serial interface for communication between the MPC5604E MCU and external devices (e.g., sensors).

The DSPI modules provide these features:

- Full duplex, three-wire synchronous transfers
- Master or slave operation
- Programmable master bit rates
- Programmable clock polarity and phase
- End-of-transmission interrupt flag
- Programmable transfer baud rate
- Programmable data frames from 4 to 16 bits
- As many as four chip select lines available per DSPI module, depending on package and pin multiplexing, enable 12 external devices to be selected using external multiplexing from a single DSPI
- Eight clock and transfer attributes registers
- Chip select strobe available as alternate function on one of the chip select pins for de-glitching
- FIFOs for buffering as many as five transfers on the transmit and receive side
- Queueing operation possible through use of the eDMA
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis
- Modified SPI transfer formats for communication with slower peripheral devices

1.8.20 Serial communication interface module (LINFlex)

The LINFlex on the MPC5604E features the following:

- Supports LIN Master mode, LIN Slave mode and UART mode
- LIN state machine compliant to LIN1.3, 2.0, and 2.1 Specifications

- Handles LIN frame transmission and reception without CPU intervention
- LIN features
 - Autonomous LIN frame handling
 - LIN0 supports master and slave mode with 16 identifier filters
 - LIN1 supports master mode only (no identifier filters required)
 - Message buffer to store Identifier and as much as 8 data bytes
 - Supports message length as long as 64 bytes
 - Detection and flagging of LIN errors: Sync field; Delimiter; ID parity; Bit; Framing; Checksum and Time-out errors
 - Classic or extended checksum calculation
 - Configurable Break duration as long as 36-bit times
 - Programmable Baud rate pre-scalers (13-bit mantissa, 4-bit fractional)
 - Diagnostic features: Loop back; Self Test; LIN bus stuck dominant detection
 - Interrupt-driven operation with 16 interrupt sources
- LIN slave mode features
 - Autonomous LIN header handling
 - Autonomous LIN response handling
- UART mode
 - Full-duplex operation
 - Standard non return-to-zero (NRZ) mark/space format
 - Data buffers with 4-byte receive, 4-byte transmit
 - Configurable word length (8-bit or 9-bit words)
 - Error detection and flagging
 - Parity, Noise and Framing errors
 - Interrupt-driven operation with four interrupt sources
 - Separate transmitter and receiver CPU interrupt sources
 - 16-bit programmable baud-rate modulus counter and 16-bit fractional
 - Two receiver wake-up methods

1.8.21 eTimer

The eTimer module provides six 16-bit general purpose up/down timer/counter.

The following features are implemented:

- Individual channel capability
 - Input capture trigger
 - Output compare
 - Double buffer (to capture rising edge and falling edge)
 - Separate pre-scaler for each counter

- Selectable clock source
- 0% to 100% pulse measurement
- Rotation direction flag (Quad decoder mode)
- Maximum count rate
- Counters are cascadeable
- Programmable count modulo
- Quadrature decode capabilities
- Counters can share available input pins
- Count once or repeatedly
- Counters are pre-loadable

1.8.22 Successive approximation Analog-to-Digital Converter (ADC)

The ADC module provides the following features:

- Analog part:
 - One on-chip AD converter
 - 10-bit AD resolution
 - Conversion time, including sampling time, less than 1 μ s (at full precision)
 - Typical sampling time is 150 ns min. (at full precision)
 - Differential non-linearity error (DNL) ± 1 LSB
 - Integral non-linearity error (INL) ± 1.5 LSB
 - TUE < 3 LSB
 - Single-ended input signal range from 0 to $V_{DD_HV_ADC}$
 - The ADC supply can be equal to $V_{DD_HV_IO}$ ($V_{DD_HV_ADC} = 3.3$ V)
 - The ADC supply and the ADC reference are not independent from each other (they are internally bonded to the same pad)
 - Sample times of 2 (default), 8, 64, or 128 ADC clock cycles
- Digital part:
 - 8 input channels
 - 4 channels routed to the pins
 - 3 internal connections: 1 \times temperature sensor, 1 \times core voltage, 1 \times IO voltage
 - Four analog watchdogs comparing ADC results against predefined levels (low, high, range) before results are stored in the appropriate ADC result location,
 - Register-based interface with the CPU: control register, status register, and one result register per channel
 - ADC state machine managing 3 request flows: regular command, hardware injected command through eTimer and software injected command
 - DMA compatible interface

1.8.23 Fault Collection Unit (FCU)

The FCU provides an independent fault reporting mechanism even when the CPU is not performing properly.

The FCU module includes following features:

- Collection of critical faults (all of these must be glitch free)
- Reporting of selected critical faults to external
- Fault flag status kept over non-destructive reset for later analysis (in a "Freeze" register)
- Continuous and synchronous latch of MC state
- MC state kept over non-destructive reset for later analysis (in a "Freeze" register)
- 4 states finite state machine (Init, Normal, Alarm, Fault)
- Different actions can be taken depending on fault type.
- Selectable protocols for fault signal indication in Fault state (dual-rail, time-switching, bi-stable)
- Programmable clock prescaler for time-switching output signal generation
- Protection mechanism to avoid un-wanted clearing of fault flags
- Internal logic testing, by using a fake fault generator during initialization phase

1.8.24 Cyclic Redundancy Check (CRC)

The CRC has the following major features:

- 2 contexts (static parameter) for the concurrent CRC computation
- Separate CRC engine for each context
- 0-wait states during the CRC computation (pipeline scheme)
- 3 hardwired polynomials (CRC-8, CRC-16-CCITT, CRC-32)
- Support for byte/half-word/word width of the input data stream

1.8.25 Video encoder

- Image resolution up to 1280×800 at 30 fps
- Low latency compression with MJPEG format
- Color sub-sampling from YUV4:2:2 to YUV4:2:0
- 8 bits per pixel component
- 12 bits per pixel component
- Support compression ratio from 1:20 to 1:5
- Support for the Ethernet Controller DMA via CPU Interrupt

1.8.26 Serial Audio Interface (SAI)

- Supports up to 6 (stereo) audio channels
- Transmitter with independent Bit Clock and Frame Sync supporting 4 data channels

- Receiver with independent Bit Clock and Frame Sync supporting 4 data channels
- Maximum Frame Size of 16 Words
- Word size of between 8-bits and 32-bits Word size configured separately for first word and remaining words in frame
- Asynchronous 8 × 32-bit FIFO for each Transmit and Receive Channel
- Restarts after FIFO Error

1.8.27 Ethernet AVB (FEC + PTP + RTC)

The Ethernet modules provide 100 Mbits/s data communication for all use cases. To support Ethernet AVB (Audio Video Bridging), this module group consists of following modules:

- FEC (Ethernet base module)
- PTP (IEEE 1588 precision time protocol)
- RTC (Real time clock required for precision time protocol)

MPC5604E does not integrate the PHY components of the Ethernet, thus, the FEC connects via the MII-Lite interface (14 pins) to the external PHY. In addition to the MII-Lite interface, the RTC provides a single timer pin that is directly linked to the precision time.

Support for different Ethernet MAC-PHY interfaces:

- 100 Mbits/s IEEE 802.3 MII-Lite 1
- 10 Mbits/s IEEE 802.3 MII-Lite
- IEEE 802.3 full duplex flow control and half duplex flow
- Programmable max frame length
- Address recognition
- Word size configured separately for first word and remaining words in frame
- Asynchronous 4 x 32-bit FIFO for each Transmit and Receive Channel
- Graceful restart after FIFO Error:
 - Frames with broadcast address may be always accepted or always rejected
 - Exact match for single 48-bit individual (unicast) address
 - Asynchronous 4 x 32-bit FIFO for each Transmit and Receive Channel
 - Hash (64-bit hash) check of individual (unicast) address
 - Hash (64-bit hash) check of group (multicast) address
 - Promiscuous mode
- Internal loop-back

1.8.27.1 Precision Time Protocol

Hardware assistance for IEEE1588 Precision Time Protocol v1.0

- Supports user configured values for PTP header fields
- Support timestamp overrun report for TX and RX

Overview

- Supports interrupts notification due to following: RX PTP frame detection, TX
- PTP frame transmission which was marked by the Software as a PTP frame, RX and TX timestamp overrun error

1.8.27.2 RTC

Support single IEEE1588 RTC

- Support timer frequency compensation
- Support timer offset update
- One 64-bit FIPER start register. Used to define the starting time of PPS signals generation
- Support timer frequency compensation
- Separate maskable timer interrupt event register
- Phase aligned adjustable (divide by N) clock output

The MPC5604E MCU tools and third-party developers are similar to those used for the Freescale MPC5500 product family, offering a widespread, established network of tool and software vendors. The device also features a high-performance Nexus debug interface.

Chapter 2 Memory Map

Table 2 shows the memory map for the MPC5604E device. All addresses on the MPC5604E, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each IP block.

Table 2. System memory map

Start Address	End Address	Size (KB)	Description
On-Chip Flash Memory (Code Flash)			
0x0000_0000	0x0000_3FFF	16	Code Flash Array 0
0x0000_4000	0x0000_7FFF	16	
0x0000_8000	0x0000_FFFF	32	
0x0001_0000	0x0000_17FF	32	
0x0001_8000	0x0001_BFFF	16	
0x0001_C000	0x0001_FFFF	16	
0x0002_0000	0x0002_FFFF	64	
0x0003_0000	0x0003_FFFF	64	
0x0004_0000	0x0005_FFFF	128	
0x0006_0000	0x0007_FFFF	128	
0x0008_0000	0x001F_FFFF	1536	
On-Chip Flash Memory (Shadow for Code Flash)			
0x0020_0000	0x0020_3FFF	16	Code Flash Array 0 Shadow Sector
0x0020_4000	0x003F_FFFF	2032	Reserved
On-Chip Flash Memory (Test Sector for Code Flash)			
0x0040_0000	0x0040_3FFF	16	Code Flash Array 0 Test Sector
0x0040_4000	0x005F_FFFF	2032	Reserved
On-Chip Flash Memory (Data Flash)			
0x0080_0000	0x0080_3FFF	16	Data Flash Array 0
0x0080_4000	0x0080_7FFF	16	Data Flash Array 0
0x0080_8000	0x0080_BFFF	16	Data Flash Array 0
0x0080_C000	0x0080_FFFF	16	Data Flash Array 0

Table 2. System memory map (continued)

Start Address	End Address	Size (KB)	Description
0x0081_0000	0x009F_FFFF	1984	Reserved
On-Chip Flash Memory (Shadow Sector for Data Flash)			
0x00A0_0000	0x00BF_FFFF	2048	Reserved
On-Chip Flash Memory (Test Sector for Data Flash)			
0x00C0_0000	0x00C0_1FFF	8	Reserved
0x00C0_2000	0x00C0_3FFF	8	Data Flash Test Sector
0x00C0_4000	0x00FF_FFFF	4080	Reserved
Emulation Mapping			
0x0100_0000	0x01FF_FFFF	524288	Reserved
SRAM			
0x4000_0000	0x4000_9FFF	40	SRAM
0x4000_A000	0x4000_FFFF	24	SRAM
0x4001_0000	0x4001_7FFF	32	SRAM
0x4001_8000	0x4FFF_FFFF	262048	Reserved
0x5000_0000	0x5000_1FFF	8	Video Output Buffer
0x5000_2000	0x5000_3FFF	8	Mirrored Video Output Buffer (from 0x50000000)
0x5000_4000	0x5FFF_FFFF	262128	Reserved
0x6000_0000	0x7FFF_FFFF	524288	Reserved

Table 3. Peripheral memory map

Start Address	End Address	Size (KB)	ME_PCTL	Description
0xC3F8_0000	0xC3F8_7FFF	32	—	Reserved
0xC3F8_8000	0xC3F8_BFFF	16	—	Code Flash Configuration 0 (CFLASH0)
0xC3F8_C000	0xC3F8_FFFF	16	—	Data Flash Configuration (DFLASH0)
0xC3F9_0000	0xC3F9_3FFF	16	—	System Integration Unit Lite (SIUL)
0xC3F9_4000	0xC3F9_7FFF	16	—	WakeUP Unit (WKUP)
0xC3F9_8000	0xC3FD_7FFF	256	—	Reserved
0xC3FD_8000	0xC3FD_BFFF	16	—	System Status and Configuration Module (SSCM)
0xC3FD_C000	0xC3FD_FFFF	16	—	Mode Entry (MC_ME)

Table 3. Peripheral memory map

Start Address	End Address	Size (KB)	ME_PCTL	Description
0xC3FE_0000	0xC3FE_3FFF	16	—	Clock related modules Note: Refer to Table 7 for details.
0xC3FE_4000	0xC3FE_7FFF	16	—	Reset Generation Module (MC_RGM)
0xC3FE_8000	0xC3FE_BFFF	16	—	Power Control Unit (MC_PCU)
0xC3FE_C000	0xC3FE_FFFF	16	—	Reserved
0xC3FF_0000	0xC3FF_3FFF	16	92	Periodic Interrupt Timer (PTI)
0xC3FF_4000	0xC3FF_FFFF	48	—	Reserved
AIPS(0) - Off Platform Peripherals				
0xFFE0_0000	0xFFE0_3FFF	16	32	Analog to Digital Converter 0 (ADC0)
0xFFE0_4000	0xFFE1_7FFF	80	—	Reserved
0xFFE1_8000	0xFFE1_BFFF	16	38	eTimer 0
0xFFE1_C000	0xFFE2_FFFF	80	—	Reserved
0xFFE3_0000	0xFFE3_3FFF	16	44	Inter IC Bus Interface Controller 0 (I2C0)
0xFFE3_4000	0xFFE3_7FFF	16	45	Inter IC Bus Interface Controller 1 (I2C1)
0xFFE3_8000	0xFFE3_FFFF	32	—	Reserved
0xFFE4_0000	0xFFE4_3FFF	16	48	LinFlex 0
0xFFE4_4000	0xFFE4_7FFF	16	49	LinFlex 1
0xFFE4_8000	0xFFE6_7FFF	128	—	Reserved
0xFFE6_8000	0xFFE6_BFFF	16	58	Cyclic Redundary Checker (CRC)
0xFFE6_C000	0xFFE6_FFFF	16	—	Fault Collection Unit (FCU)
0xFFE7_0000	0xFFE7_3FFF	16	—	Reserved
0xFFE7_4000	0xFFE7_7FFF	16	61	Precision Time Stamps (PTP)
0xFFE7_8000	0xFFE7_BFFF	16	62	Real-Time Counter (CE_RTC)
0xFFE7_C000	0xFFE7_FFFF	80	—	Reserved
AIPS(0) - Off Platform Peripherals (mirrored from AIPS(1) range 0xC3F80000 - 0xC3FFFFFF)				
0xFFE8_0000	0xFFEF_FFFF	512	—	Mirrored
AIPS(0) - On Platform Peripherals				
0xFFFF_0000	0xFFFF_7FFF	224	—	Reserved
0xFFFF_8000	0xFFFF_BFFF	16	—	Software Watchdog Timer 0 (SWT0)
0xFFFF_C000	0xFFFF_FFFF	16	—	System Timer Module 0 (STM0)
0xFFFF4_0000	0xFFFF4_3FFF	16	—	Miscellaneous Control Module (MCM)

Table 3. Peripheral memory map

Start Address	End Address	Size (KB)	ME_PCTL	Description
0xFFFF4_4000	0xFFFF4_7FFF	16	—	Direct Memory Access Controller 0 (DMA2x)
0xFFFF4_8000	0xFFFF4_BFFF	16	—	Interrupt Controller (INTC)
AIPS(0) - Off Platform Peripherals				
0xFFFF4_C000	0xFFFF4_FFFF	16	—	Ethernet (FEC)
0xFFFF5_0000	0xFFFF7_FFFF	192	—	Reserved
0xFFFF8_0000	0xFFFF8_FFFF	64	—	Reserved
0xFFFF9_0000	0xFFFF9_3FFF	16	4	DSPI 0
0xFFFF9_4000	0xFFFF9_7FFF	16	5	DSPI 1
0xFFFF9_8000	0xFFFF9_BFFF	16	6	DSPI 2
0xFFFF9_C000	0xFFFFB_FFFF	144	—	Reserved
0xFFFFC_0000	0xFFFFC_3FFF	16	16	FlexCan 0 (CAN0)
0xFFFFC_4000	0xFFFFD_7FFF	80	—	Reserved
0xFFFFD_8000	0xFFFFD_BFFF	16	22	SAI 0
0xFFFFD_C000	0xFFFFD_FFFF	16	23	DMA Channel Multiplexer (DMA_CH_MUX)
0xFFFFE0000	0xFFFFE_FFFF	64	—	Reserved
0xFFFFF0000	0xFFFFF_3FFF	16	28	SAI 1
0xFFFFF4000	0xFFFFF_7FFF	16	29	SAI 2
0xFFFFF8000	0xFFFFF_BFFF	16	30	Video Data Path/Parallel Digital Interface (PDI)
0xFFFFFC000	0xFFFFF_FFFF	16	—	Boot Assist Module (BAM)

Chapter 3 Signal Description

3.1 Introduction

This chapter describes signals that connect off-chip. It includes a signal properties summary, power and ground segmentation summary, package pinouts, and detailed descriptions of signals. Because the MPC5604E comes in multiple packages, some signals may not be available on every package. Refer to the *MPC5604E Microcontroller Data Sheet* for electrical characteristics.

3.2 Signal Properties Summary

[Table 4](#) shows the signals properties for each pin on MPC5604E. For all port pins that have an associated SIU_PCR n register to control pin properties, the supported functions column lists the functions associated with the programming of the SIU_PCR n [PA] bit in the order: general-purpose input/output (GPIO), function 1, function 2, and function 3. When an alternate function is not implemented for a value of SIU_PCR n [PA], a dash is shown in the Description column and the respective value in the PA bitfield is reserved.

Table 4. Pin muxing

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
Port A (16-bit)									
A[0]	PCR[0]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[0] D[0] — — D[11] SIN EIRQ[0]	SIUL SAI0 — — VID DSPI 1 SIUL	I/O I/O — — I I I	Slow	Medium	2	2
A[1]	PCR[1]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[1] D[1] SOUT — D[10] EIRQ[1]	SIUL SAI0 DSPI1 — VID SIUL	I/O I/O O — I I	Slow	Medium	3	4
A[2]	PCR[2]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[2] D[2] SCK D[0] D[9] ETC[5] EIRQ[2]	SIUL SAI0 DSPI1 SAI1 VID ETIMER0 SIUL	I/O I/O I/O I/O I I I	Slow	Medium	4	6
A[3]	PCR[3]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[3] D[3] — D[0] D[8] SIN EIRQ[3]	SIUL SAI0 — SAI2 VID DSPI2 SIUL	I/O I/O — I/O I I I	Slow	Medium	5	8
A[4]	PCR[4]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[4] SYNC SOUT — D[7] ETC[3] EIRQ[4]	SIUL SAI0 DSPI2 — VID ETIMER0 SIUL	I/O I/O O — I I I	Slow	Medium	8	15
A[5]	PCR[5]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[5] SYNC SCK D[0] CLK ETC[4] EIRQ[5]	SIUL SAI1 DSPI2 SAI1 VID ETIMER0 SIUL	I/O I/O I/O I/O I I I	Medium	Fast	9	16

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
A[6]	PCR[6]	ALT0 ALT1 ALT2 ALT3 — — — —	GPIO[6] SYNC CS0 — VSYNC D[0] ETC[1] EIRQ[6]	SIUL SAI2 DSPI2 — VID VID ETIMER0 SIUL	I/O I/O I/O — I I I I	Slow	Medium	10	17
A[7]	PCR[7]	ALT0 ALT1 ALT2 ALT3 — — — —	GPIO[7] BCLK CS1 — HREF D[1] ETC[2] EIRQ[7]	SIUL SAI0 DSPI2 — VID VID ETIMER0 SIUL	I/O I/O I/O — I I I I	Slow	Medium	16	23
A[8]	PCR[8]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[8] BCLK CS0 D[0] D[6] RX EIRQ[8]	SIUL SAI1 DSPI1 SAI2 VID LIN1 SIUL	I/O I/O I/O I/O I I I	Slow	Medium	24	37
A[9]	PCR[9]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[9] BCLK CS1 TX D[5] EIRQ[9]	SIUL SAI2 DSPI1 LIN1 VID SIUL	I/O I/O I/O O I I	Slow	Medium	25	38
A[10]	PCR[10]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[10] MCLK ETC[5] — D[4] SIN EIRQ[10]	SIUL SAI2 ETIMER0 — VID DSPI0 SIUL	I/O I/O I/O — I I I	Slow	Medium	26	39
A[11]	PCR[11]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[11] TX CS1 CS0 D[3] RX RX	SIUL CAN0 DSPI0 DSPI1 VID LIN0 LIN1	I/O O O I/O I I I	Slow	Medium	27	40

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
A[12]	PCR[12]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[12] TX CS0 TX D[2] RX EIRQ[11]	SIUL LIN0 DSPI0 LIN1 VID CAN0 SIUL	I/O O I/O O I I I	Slow	Medium	28	41
A[13]	PCR[13]	ALT0 ALT1 ALT2 ALT3 —	GPIO[13] CLK F[0] CS0 EIRQ[12]	SIUL IIC1 FCU0 DSPI0 SIUL	I/O I/O O I/O I	Slow	Medium	29	42
A[14]	PCR[14]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[14] DATA F[1] CS1 SIN EIRQ[13]	SIUL IIC1 FCU0 DSPI0 DSPI0 SIUL	I/O I/O O O I I	Slow	Medium	30	43
A[15]	PCR[15]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[15] SCK PPS3 MCLK SCK ETC[0] EIRQ[18]	SIUL DSPI0 CE_RTC SAI1 DSPI1 ETIMER0 SIUL	I/O I/O O I/O I I I	Slow	Medium	61	95
Port B (16-bit)									
B[0]	PCR[16]	ALT0 ALT1 ALT2 ALT3 —	GPIO[16] TX ALARM2 BCLK AN[0]	SIUL CAN0 CE_RTC SAI1 ADC0 ⁸	I/O O O I/O I	Slow	Medium	17	26
B[1]	PCR[17]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[17] — — D[0] AN[1] RX TRIGGER2	SIUL — — SAI1 ADC0 ⁸ CAN0 CE_RTC	I/O — — I/O I I I	Slow	Medium	18	27
B[2]	PCR[18]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[18] TX PPS2 ALARM1 AN[2] TRIGGER1	SIUL LIN0 CE_RTC CE_RTC ADC0 ⁸ CE_RTC	I/O O O O I I	Slow	Medium	19	28

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
B[3]	PCR[19]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[19] ETC[2] SOUT PPS1 AN[3] RX EIRQ[14]	SIUL ETIMER0 DSPI0 CE_RTC ADC0 ⁸ LIN0 SIUL	I/O I/O I/O O I I I	Slow	Medium	20	29
B[4]	PCR[20]	ALT0 ALT1 ALT2 ALT3 —	GPI[20] — — — RX_DV	SIUL — — — FEC	I — — — I	Slow	Medium	32	50
B[5]	PCR[21]	ALT0 ALT1 ALT2 ALT3	GPIO[21] TX_D0 DEBUG[0] —	SIUL FEC SSCM —	I/O O I/O —	Slow	Medium	33	55
B[6]	PCR[22]	ALT0 ALT1 ALT2 ALT3	GPIO[22] TX_D1 DEBUG[1] —	SIUL FEC SSCM —	I/O O I/O —	Slow	Medium	34	56
B[7]	PCR[23]	ALT0 ALT1 ALT2 ALT3	GPIO[23] TX_D2 DEBUG[2] —	SIUL FEC SSCM —	I/O O I/O —	Slow	Medium	39	62
B[8]	PCR[24]	ALT0 ALT1 ALT2 ALT3	GPIO[24] TX_D3 DEBUG[3] —	SIUL FEC SSCM —	I/O O I/O —	Slow	Medium	44	67
B[9]	PCR[25]	ALT0 ALT1 ALT2 ALT3	GPIO[25] TX_EN DEBUG[4] —	SIUL FEC SSCM —	I/O O I/O —	Slow	Medium	45	70
B[10]	PCR[26]	ALT0 ALT1 ALT2 ALT3	GPIO[26] MDC DEBUG[5] —	SIUL FEC SSCM —	I/O O I/O —	Slow	Medium	46	73
B[11]	PCR[27]	ALT0 ALT1 ALT2 ALT3	GPIO[27] MDIO DEBUG[6] —	SIUL FEC SSCM —	I/O I/O I/O —	Slow	Medium	48	75
B[12]	PCR[28]	ALT0 ALT1 ALT2 ALT3 —	GPIO[28] — DEBUG[7] — TX_CLK	SIUL — SSCM — FEC	I/O — I/O — I	Slow	Medium	49	76

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
B[13]	PCR[29]	ALT0 ALT1 ALT2 ALT3 —	GPI[29] — — — RX_D0	SIUL — — — FEC	I — — — I	Slow	Medium	50	77
B[14]	PCR[30]	ALT0 ALT1 ALT2 ALT3 —	GPI[30] — — — RX_D1	SIUL — — — FEC	I — — — I	Slow	Medium	51	79
B[15]	PCR[31]	ALT0 ALT1 ALT2 ALT3 —	GPI[31] — — — RX_D2	SIUL — — — FEC	I — — — I	Slow	Medium	52	81
Port C (64-pin: 7-bit; 100-pin: 16-bit)									
C[0]	PCR[32]	ALT0 ALT1 ALT2 ALT3 —	GPI[32] — — — RX_D3	SIUL — — — FEC	I — — — I	Slow	Medium	53	82
C[1]	PCR[33]	ALT0 ALT1 ALT2 ALT3 — —	GPI[33] — — — RX_CLK EIRQ[15]	SIUL — — — FEC SIUL	I — — — I I	Slow	Medium	54	83
C[2]	PCR[34]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[34] ETC[0] TX PPS1 D[0] RX EIRQ[16]	SIUL ETIMER0 CAN0 CE_RTC VID LIN0 SIUL	I/O I/O O O I I I	Slow	Medium	57	91
C[3]	PCR[35]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[35] ETC[1] TX SYNC D[1] RX EIRQ[17]	SIUL ETIMER0 LIN0 SAI1 VID CAN0 SIUL	I/O I/O O I/O I I I	Slow	Medium	60	94

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
C[4]	PCR[36]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[36] CLK_OUT ETC[4] MCLK TRIGGER1 ABS[0] EIRQ[19]	SIUL MC_CGL ETIMER0 SAI0 CE_RTC MC_RGM SIUL	I/O O I/O I/O I I I	Medium	Fast	62	96
C[5]	PCR[37]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[37] CLK ETC[3] CS2 ABS[2] EIRQ[20]	SIUL IIC0 ETIMER0 DSPI2 MC_RGM SIUL	I/O — I/O O I I	Slow	Medium	63	99
C[6]	PCR[38]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[38] DATA CS0 CS3 FAB EIRQ[21]	SIUL IIC0 DSPI1 DSPI2 MC_RGM SIUL	I/O — I/O O I I	Slow	Medium	64	100
C[7]	PCR[39]	ALT0 ALT1 ALT2 ALT3 —	GPIO[39] TXD — — RXD	SIUL LIN0 — — LIN1	I/O O — — I	Slow	Medium	—	3
C[8]	PCR[40]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[40] TXD — — RXD EIRQ[22]	SIUL LIN1 — — LIN0 SIUL	I/O O — — I I	Slow	Medium	—	5
C[9]	PCR[41]	ALT0 ALT1 ALT2 ALT3 — —	GPI[41] — — — SIN EIRQ[23]	SIUL — — — DSPI0 SIUL	I — — — I I	Slow	Medium	—	7
C[10]	PCR[42]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[42] ETC[5] ETC[4] — SIN EIRQ[24]	SIUL ETIMER0 ETIMER0 — DSPI1 SIUL	I/O I/O I/O — I I	Slow	Medium	—	24
C[11]	PCR[43]	ALT0 ALT1 ALT2 ALT3	GPIO[43] ETC[2] ETC[1] ETC[3]	SIUL ETIMER0 ETIMER0 ETIMER0	I/O I/O I/O I/O	Slow	Medium	—	25

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
C[12]	PCR[44]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[44] PPS1 PPS2 ALARM1 TRIGGER1 TRIGGER2 EIRQ[25]	SIUL CE_RTC CE_RTC CE_RTC CE_RTC CE_RTC SIUL	I/O O O O I I I	Slow	Medium	—	44
C[13]	PCR[45]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[45] — — — D[1] EIRQ[26]	SIUL — — — VID SIUL	I/O — — — I I	Slow	Medium	—	46
C[14]	PCR[46]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[46] — — — D[0] EIRQ[27]	SIUL — — — VID SIUL	I/O — — — I I	Slow	Medium	—	47
C[15]	PCR[47]	ALT0 ALT1 ALT2 ALT3 —	GPI[47] — — — COL	SIUL — — — FEC	I — — — I	Slow	Medium	—	48
Port D (100-pin package: 16-bit)									
D[0]	PCR[48]	ALT0 ALT1 ALT2 ALT3	GPIO[48] MDO0 — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	9
D[1]	PCR[49]	ALT0 ALT1 ALT2 ALT3	GPIO[49] MCK0 — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	14
D[2]	PCR[50]	ALT0 ALT1 ALT2 ALT3	GPIO[50] EVTO — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	13
D[3]	PCR[51]	ALT0 ALT1 ALT2 ALT3	GPIO[51] MSEO1 — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	72
D[4]	PCR[52]	ALT0 ALT1 ALT2 ALT3	GPIO[52] MSEO0 — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	78

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
D[5]	PCR[53]	ALT0 ALT1 ALT2 ALT3	GPIO[53] MDO3 — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	80
D[6]	PCR[54]	ALT0 ALT1 ALT2 ALT3	GPIO[54] MDO2 — —	SIUL NEXUS — —	I/O O — —	Slow	Medium	—	84
D[7]	PCR[55]	ALT0 ALT1 ALT2 ALT3	GPIO[55] MDO1 — —	SIUL NEXUS — —	I/O — — —	Slow	Medium	—	98
D[8]	PCR[56]	ALT0 ALT1 ALT2 ALT3 —	GPI[56] — — — — EVTI	SIUL — — — — NEXUS	I — — — — I	Slow	Medium	—	10
D[9]	PCR[57]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[57] ETC[3] ETC[2] — RXD EIRQ[28]	SIUL ETIMER0 ETIMER0 — CAN0 SIUL	I/O I/O I/O — I I	Slow	Medium	—	49
D[10]	PCR[58]	ALT0 ALT1 ALT2 ALT3	GPIO[58] TXD — —	SIUL CAN0 — —	I/O O — —	Slow	Medium	—	51
D[11]	PCR[59]	ALT0 ALT1 ALT2 ALT3	GPIO[59] ETC[0] ETC[5] ETC[4]	SIUL ETIMER0 ETIMER0 ETIMER0	I/O I/O I/O I/O	Slow	Medium	—	52
D[12]	PCR[60]	ALT0 ALT1 ALT2 ALT3 —	GPIO[60] ETC[1] ETC[0] — SIN	SIUL ETIMER0 ETIMER0 — DSPI0	I/O I/O I/O — I	Slow	Medium	—	53
D[13]	PCR[61]	ALT0 ALT1 ALT2 ALT3 — —	GPI[61] — — — — CRS EIRQ[29]	SIUL — — — — FEC SIUL	I — — — — I I	Slow	Medium	—	54

Table 4. Pin muxing (continued)

Port pin	PCR register	Alternate function ^{1,2,8}	Functions	Peripheral ³	I/O direction ⁴	Pad speed ⁵		Pin ⁶	
						SRC = 0	SRC = 1	64-pin	100-pin ⁷
D[14]	PCR[62]	ALT0 ALT1 ALT2 ALT3 — —	GPI[62] — — — RX_ER EIRQ[30]	SIUL — — — FEC SIUL	I — — — I I	Slow	Medium	—	57
D[15]	PCR[63]	ALT0 ALT1 ALT2 ALT3	GPIO[63] F[0] — —	SIUL FCU0 — —	I/O O — —	Slow	Medium	—	69
Port E (100-pin package: 7-bit)									
E[0]	PCR[64]	ALT0 ALT1 ALT2 ALT3	GPIO[64] F[1] — —	SIUL FCU0 — —	I/O O — —	Slow	Medium	—	68
E[1]	PCR[65]	ALT0 ALT1 ALT2 ALT3	GPIO[65] TX_ER — —	SIUL FEC — —	I/O O — —	Slow	Medium	—	71
E[2]	PCR[66]	ALT0 ALT1 ALT2 ALT3 — —	GPI[66] — — — RXD EIRQ[31]	SIUL — — — LIN1 SIUL	I — — — I I	Slow	Medium	—	85
E[3]	PCR[67]	ALT0 ALT1 ALT2 ALT3	GPIO[67] TXD — —	SIUL LIN1 — —	I/O O — —	Slow	Medium	—	86
E[4]	PCR[68]	ALT0 ALT1 ALT2 ALT3	GPIO[68] CS0 CS0 CS0	SIUL DSPI0 DSPI1 DSPI2	I/O I/O I/O I/O	Slow	Medium	—	89
E[5]	PCR[69]	ALT0 ALT1 ALT2 ALT3	GPIO[69] SCK SCK SCK	SIUL DSPI0 DSPI1 DSPI2	I/O I/O I/O I/O	Slow	Medium	—	90
E[6]	PCR[70]	ALT0 ALT1 ALT2 ALT3 — — —	GPIO[70] SOUT SOUT SOUT SIN SIN SIN	SIUL DSPI0 DSPI1 DSPI2 DSPI0 DSPI2 DSPI2	I/O O O O I I I	Slow	Medium	—	97

- ¹ ALT0 is the primary (default) function for each port after reset.
- ² Alternate functions are chosen by setting the values of the PCR.PA bitfields inside the SIU module. PCR.PA = 00 → ALT0; PCR.PA = 01 → ALT1; PCR.PA = 10 → ALT2; PCR.PA = 11 → ALT3. This is intended to select the output functions; to use one of the input functions, the PCR.IBE bit must be written to '1', regardless of the values selected in the PCR.PA bitfields. For this reason, the value corresponding to an input only function is reported as "—".
- ³ Module included on the MCU.
- ⁴ Multiple inputs are routed to all respective modules internally. The input of some modules must be configured by setting the values of the PSMIO.PADSELx bitfields inside the SIUL module.
- ⁵ Programmable via the SRC (Slew Rate Control) bits in the respective Pad Configuration Register.
- ⁶ Additional board pull resistors are recommended when JTAG pins are not being used on the board or application.
- ⁷ The 100-pin package is not a production package. It is used for software development only.
- ⁸ Do not use ALT multiplexing when ADC channels are used.

3.3 Supply pins

Table 5 lists the supply pins for the MPC5604E.

Table 5. Supply pins

Supply			Pin	
Port Pin	Multi-bonded Power Supplies/Ground	Description	64-pin	100-pin ¹
VREG control and power supply pins. Pins available on 64-pin and 100-pin package.				
V _{DD_HV_S_BALLAST}	V _{DD_HV_S_BALLAST0}	Ballast Source/Supply Voltage	23	34
	V _{DD_HV_S_BALLAST1}	Ballast Source/Supply Voltage		
ADC0 reference and supply voltage. Pins available on 64-pin and 100-pin package.				
V _{DD_HV_ADC}	V _{DD_HV_ADC0}	ADC0 supply voltage with respect to ground (V _{SS_HV_ADC})	21	30
	V _{DD_HV_ADC0}	ADC0 high reference voltage with respect to ground (V _{SS_HV_ADC})		
V _{SS_HV_ADC}	V _{SS_HV_ADC0}	ADC0 ground voltage with respect to ground	22	31
	V _{SS_HV_ADC0}	ADC0 low reference voltage with respect to ground		

Table 5. Supply pins (continued)

Supply			Pin	
Port Pin	Multi-bonded Power Supplies/Ground	Description	64-pin	100-pin ¹
Power supply pins (3.3 V). Pins available on 64-pin and 100-pin package.				
V _{DD_HV}	V _{DD_HV_IO0_0}	Input/output ground voltage	11	18
	V _{DD_HV_OSC0}	Crystal oscillator amplifier supply voltage		
	V _{DD_HV_IO0_2}	3.3 V Input/Output Supply Voltage (supply)	38	61
	V _{DD_HV_FL01}	Code and data flash supply voltage		
	V _{DD_HV_IO0_3}	3.3 V Input/Output Supply Voltage (supply)	55	87
	V _{DD_HV_FL00}	Code and data flash supply voltage		
	V _{DD_HV}	HV Supply	-	36
V _{SS_HV}	V _{SS_HV_IO0_0}	Input/output ground voltage	12	19
	V _{SS_HV_OSC0}	Crystal oscillator amplifier ground		
	V _{SS_HV_IO0_2}	Input/output ground voltage	37	60
	V _{SS_HV_FL01}	Code and data flash supply ground		
	V _{SS_IO0_4}	Input/output ground voltage	56	35
	V _{SS_HV_FL00}	Code and data flash supply voltage		88
	V _{SS_HV}	HV Ground	47	74
Power supply pins (1.2 V). Pins available on 64-pin and 100-pin package.				
V _{DD_LV}	V _{DD_LV_COR0_3}	1.2 V supply pins for core logic and code Flash. Decoupling capacitor must be connected between these pins and the nearest V _{SS_LV_COR0_3} pin.	7	12
	V _{DD_LV_PLL0}	1.2 V PLL supply voltage		
	V _{DD_LV_COR0_2}	1.2 V supply pins for core logic and code Flash. Decoupling capacitor must be connected between these pins and the nearest V _{SS_LV_COR0_2} pin.	58	92
	V _{DD_LV_FL00}	Code and data flash supply voltage		
	V _{DD_LV_COR0_1}	1.2 V supply pins for core logic and code Flash. Decoupling capacitor must be connected between these pins and the nearest V _{SS_LV_COR0_1} pin.	35	58
	V _{DD_LV_FL01}	Code and data flash supply voltage		
	V _{DD_LV}	Core supply	-	33

Table 5. Supply pins (continued)

Supply			Pin	
Port Pin	Multi-bonded Power Supplies/Ground	Description	64-pin	100-pin ¹
V _{SS_LV}	V _{SS_LV_COR0_3}	1.2 V supply pins for core logic and code Flash. Decoupling capacitor must be connected between these pins and the nearest V _{DD_LV_COR0_3} pin.	6	11
	V _{SS_LV_PLL0}	PLL supply ground		
	V _{SS_LV_COR0_2}	1.2 V supply pins for core logic and code Flash. Decoupling capacitor must be connected between these pins and the nearest V _{DD_LV_COR0_2} pin.	59	93
	V _{SS_LV_FL0}	Code and data flash supply ground		
	V _{SS_LV_COR0_1}	1.2 V supply pins for core logic and data Flash. Decoupling capacitor must be connected between these pins and the nearest V _{DD_LV_COR0_1} pin.	36	59
	V _{SS_LV_FL1}	Code and data flash supply ground		
	V _{SS_LV}	Core ground	-	32

¹ The 100-pin package is not a production package. It is used for software development only.

3.4 System pins

Table 6 lists the system pins for the MPC5604E.

Table 6. System pins

Symbol	Description	Direction	Pad speed ¹		Pin	
			SRC = 0	SRC = 1	64-pin	100-pin ₂
Dedicated pins						
NMI	Non-maskable Interrupt	Input only	Slow	—	1	1
XTAL	Oscillator amplifier output	Output only	—	—	13	20
EXTAL	Input for oscillator amplifier circuit and internal clock generator	Input only	—	—	14	21
TDI ³	JTAG test data input	Input only	Slow	Medium	40	63
TMS ³	JTAG state machine control	Input only	Slow	Medium	41	64

Table 6. System pins (continued)

Symbol	Description	Direction	Pad speed ¹		Pin	
			SRC = 0	SRC = 1	64-pin	100-pin ₂
TCK ³	JTAG clock	Input only	Slow	—	42	65
TDO ³	JTAG test data output	Output only	Slow	Medium	43	66
Reset pin						
$\overline{\text{RESET}}$	Bidirectional reset with Schmitt trigger characteristics and noise filter	Bidirectional	Medium	—	15	22
POR_B	Power-on reset	Input only	—	—	31	45

¹ SRC values refer to the value assigned to the Slew Rate Control bits of the pad configuration register.

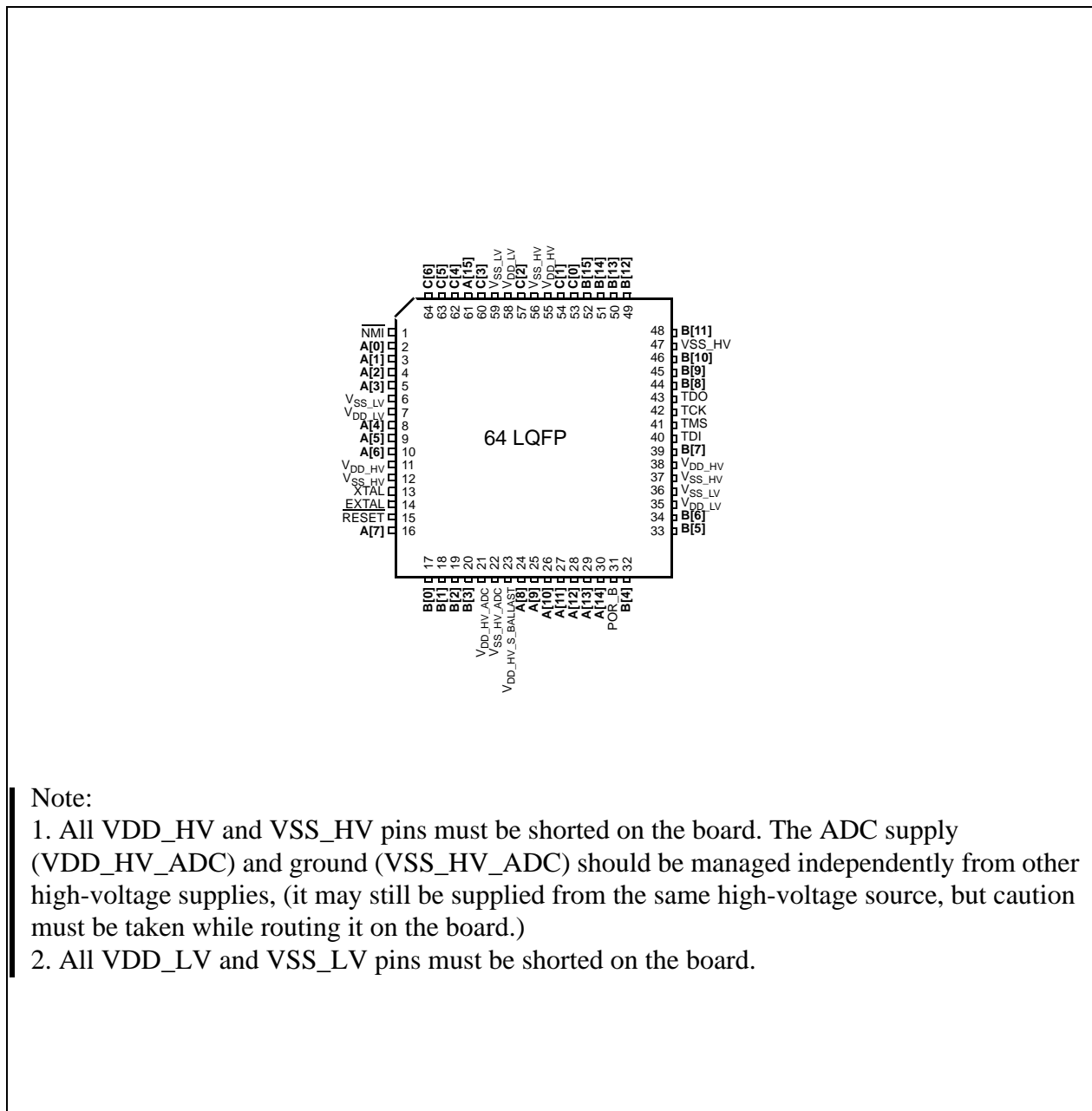
² The 100-pin package is not a production package. It is used for software development only.

³ Additional board pull resistors are recommended when JTAG pins are not being used on the board or application.

3.5 Pinouts

Figure 6 shows the 64-pin LQFP pin assignments. Figure 7 shows the 100-pin LQFP¹ pin assignments. For more information, see the *MPC5604E Microcontroller Data Sheet*.

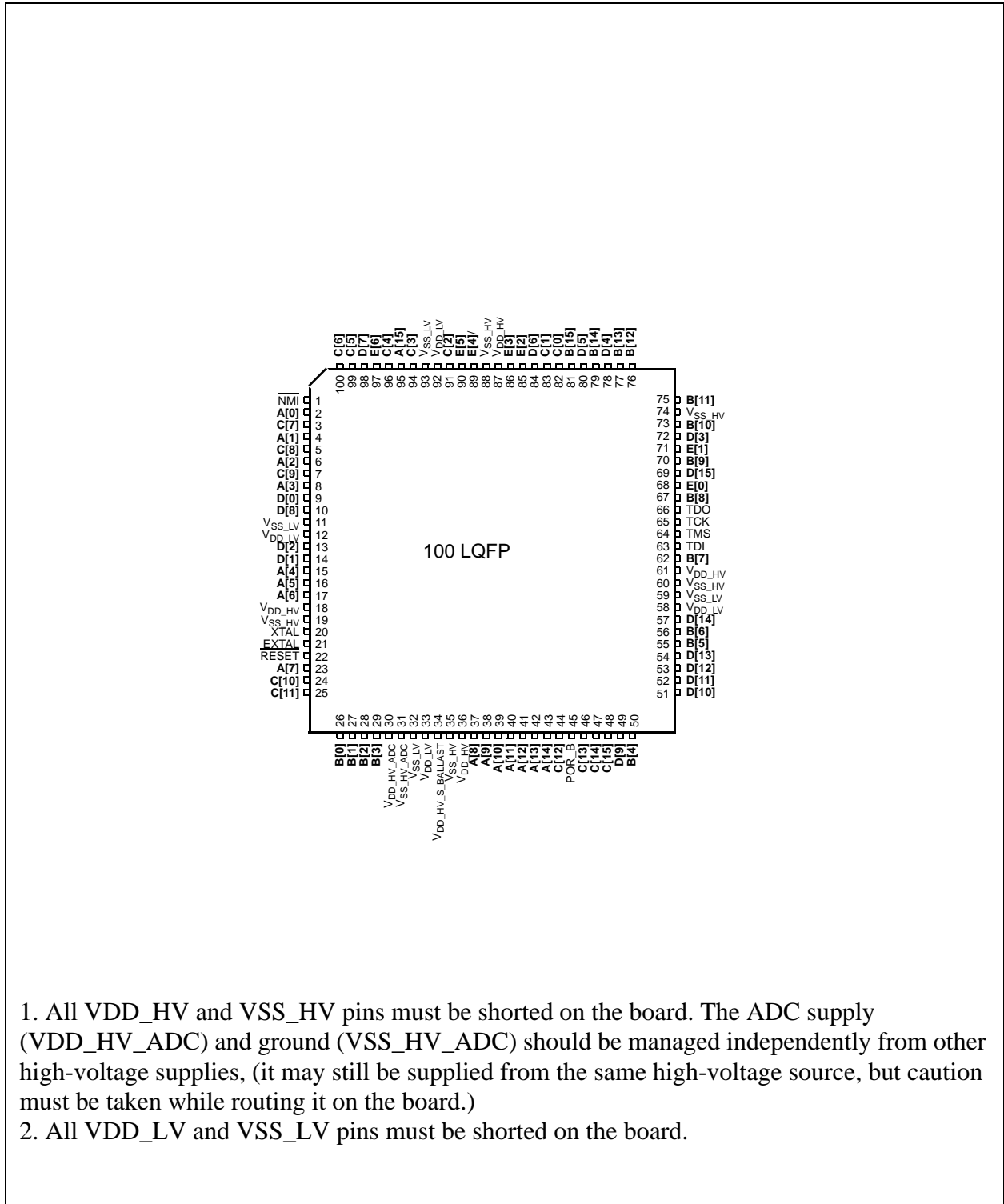
1. The 100LQFP package is not available in production quantity—it is for development only.



Note:

1. All VDD_HV and VSS_HV pins must be shorted on the board. The ADC supply (VDD_HV_ADC) and ground (VSS_HV_ADC) should be managed independently from other high-voltage supplies, (it may still be supplied from the same high-voltage source, but caution must be taken while routing it on the board.)
2. All VDD_LV and VSS_LV pins must be shorted on the board.

Figure 6. 64-pin LQFP pinout (top view)



1. All VDD_HV and VSS_HV pins must be shorted on the board. The ADC supply (VDD_HV_ADC) and ground (VSS_HV_ADC) should be managed independently from other high-voltage supplies, (it may still be supplied from the same high-voltage source, but caution must be taken while routing it on the board.)
2. All VDD_LV and VSS_LV pins must be shorted on the board.

Figure 7. 100-pin LQFP pinout (top view)¹

1. The 100-pin package is not a production package. It is used for software development only.

Chapter 4

Clock Architecture

The goal of this section is to provide designers and users with documentation to help them understand the programming model of the IC clock distribution. The following information is included:

- Clock sources
- Clock selection architecture
- Clock distribution
- Power modes

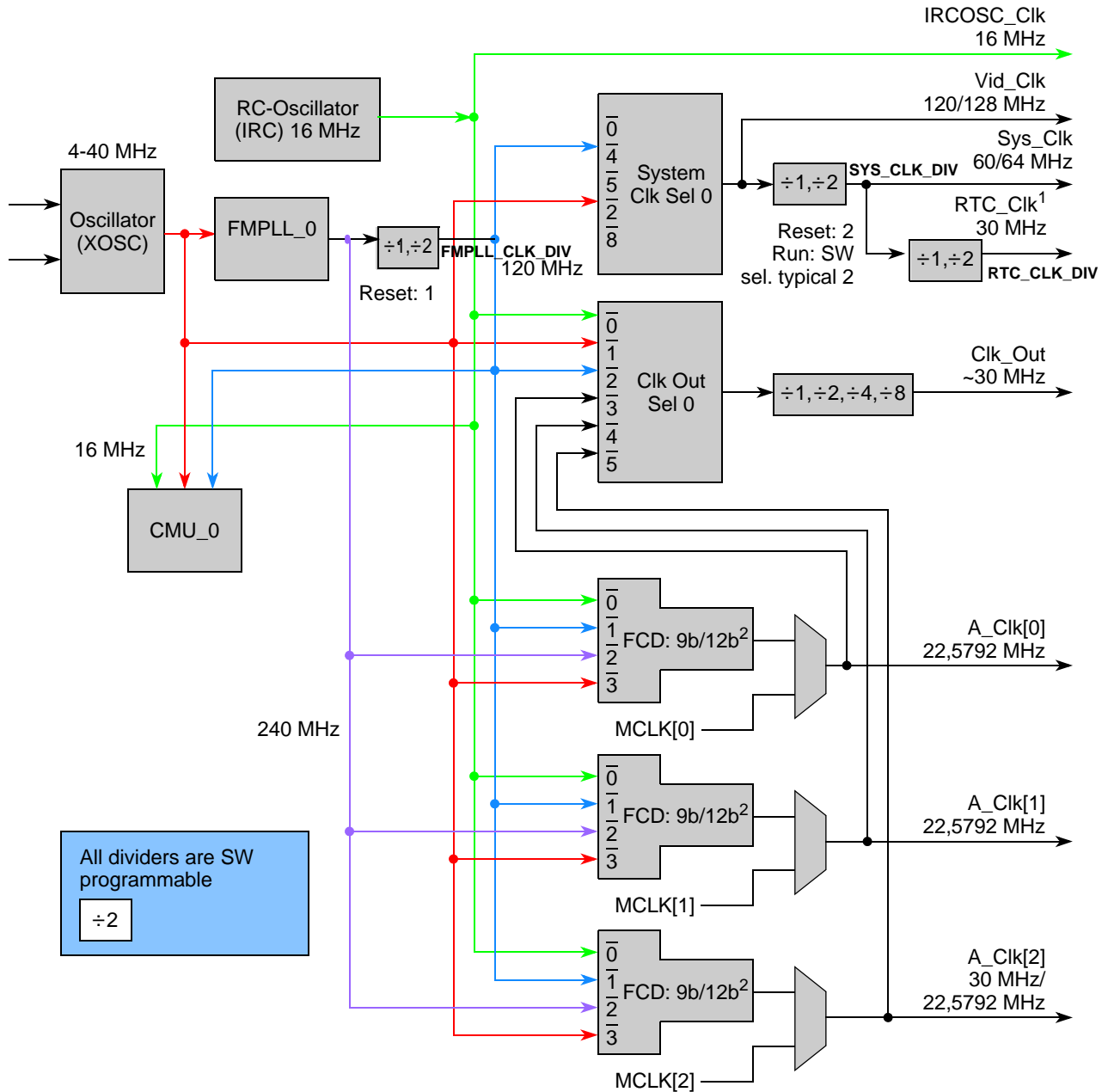
4.1 Clock related modules

The following clock related modules are instantiated on MPC5604E devices.

- 1 x Clock, Reset and Mode Handling (MagicCarpet)
- 1 x High Frequency Oscillator (XOSC)
- 1 x High Frequency RC-Oscillator (IRCOSC)
- 1 x Frequency Modulated Phase Lock Loop (FMPLL_0)
- 1 x Clock Monitoring Unit (CMU_0)
- 3 x Fractional Clock Dividers (wrapped to include AUX Clock Selectors)
- 4 x Integer Clock Dividers

4.2 High-level block diagrams

This section shows the block diagrams for the clock selection, [Figure 8](#), and for the clock distribution, [Figure 9](#), and [Figure 10](#).



Note: The maximum frequency supported by clk_out pad is 32 MHz.

1. RTC_Clk divider is SW programmable. Reset value of this divider is 2. It is requirement in RTC that ipg_clk should be half of ipg_ce_clk. So, the SW should never program this divider = 1.
2. For details, refer to MCLK Divide Register (I2S_MDR) of the Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI) chapter.

Figure 8. MPC5604E Clock Selection

The block shows a set of clock selectors and their possible inputs sources. The internal RC-Oscillator is a reliable clock source, that run independently from any other components. It is the main clock source during boot or in case of lock loss in the PLL. During Run-Mode the PLL is assumed to be used as main clock source allowing to operate the system at high frequencies. The PLL clock is derived from the Oscillator (XOSC) output. The XOSC module requires an external oscillator source, either a crystal or an external clock generator.

The main clock is the system clock `Sys_Clk`. It drives the CPU core, the crossbar, and the IPS peripheral bus including all peripheral interfaces of the IP blocks. A typical frequency would be 60MHz or 64MHz. To support the external clock frequency required for the MIILite interface of the Ethernet module (FEC) the system clock needs to higher than 50MHz. The system clock is furthermore divided by 2 and provided as possible clock for the real-time counter used to for precision timestamps (`RTC_Clk`).

The MPC5604E allows to supply for different use cases closed loop controlled clocks. For this purpose, the MPC5604E instantiates Fractional Clock Dividers (FCD), for which the frequency can be adjusted with fine granularity. To achieve a minimum clock jitter, the input frequency shall be selected as high as possible, up to the maximum input frequency the FCDs can work on. To create a closed loop control, the real clock frequency can be measured using an eTimer input, by counting the clock edges (up to `Sys_Clk/2`). Thus, low long term drift can be achieved. The FCD clocks can be provided to:

- `Clk_Out` (to a pin, e.g., used as camera clock)
- `A0_Clk` - `A2_Clk` (audio bit clocks)

For further information about FCD input clock selection, see [Section 23.1.3.1, “SAI/I2S Clock Selection” of Chapter 23, "SAI Instantiation"](#).

The MPC5604E allows to clock external components, e.g., the camera sensor. For this purpose the MPC5604E has an output pin (`Clk_Out`). The clock source for this pin can be selected using the `Clk_Out_Sel_0` selector. If this clock needs to have a low long term drift, it can be selected from the `A0/1/2_Clk`.

The `Vid_Clk` drives the video encoder module. This frequency needs to be higher than the pixel clock provided by the camera sensor. The camera provides a pixel clock up to 80MHz or 100MHz, thus, typically the `Vid_Clk` will be configured to 120MHz.

The external audio interfaces run also on dedicated frequencies that is different from the system clock. This clock can be supplied either from external or from internal. In both cases the frequency of the audio bit clock need to be closed loop controlled to remove long term drift and to prevent buffer overflows or underruns. In case the audio bit clock is provided from internal, the fractional clock divider clock needs to be used. However, this is an easy solution, the fractional clock divider implies a peak jitter from $1/2 \times$ input clock (e.g., 120MHz). Thus, the input clock for the FCD is preferably selected with a frequency as high as possible. Thus, the input selector can also select from the PLL output directly, which is running at $4 \times$ `Sys_Clk`. Because the system clock shall not below 50MHz (due to the FEC minimum clock) the FCD should operate at input frequencies of 200MHz and above. The FCD clocks can be used for low cost audio. For high quality audio an external audio clock should be used with a jitter about 100ps RMS.

[Figure 9](#) shows how the clocks from the clock selectors are connected to the system components. All BIU (bus interface units to the IPS peripheral bus) are running on system clock. The IPS clock running on `Sys_Clk` is only active on active bus cycles.

Thus, all modules also have a module clock that is typically a gated (per module) version of the system clock Sys_Clk. Besides this, the software watchdog timer SWT uses the more reliable internal RC oscillator clock IRCOSC_CLK. This clock is almost always on and thus well apt for safety functions.

The IRCOSC_Clk is also used for safety relevant functions in the Fault Collection Unit (FCU). Furthermore, this clock is used for the digital filters to detect external interrupt events. Thus, the system can resume from STOP mode, while the system clock is gated off and only the IRCOSC_Clk is running.

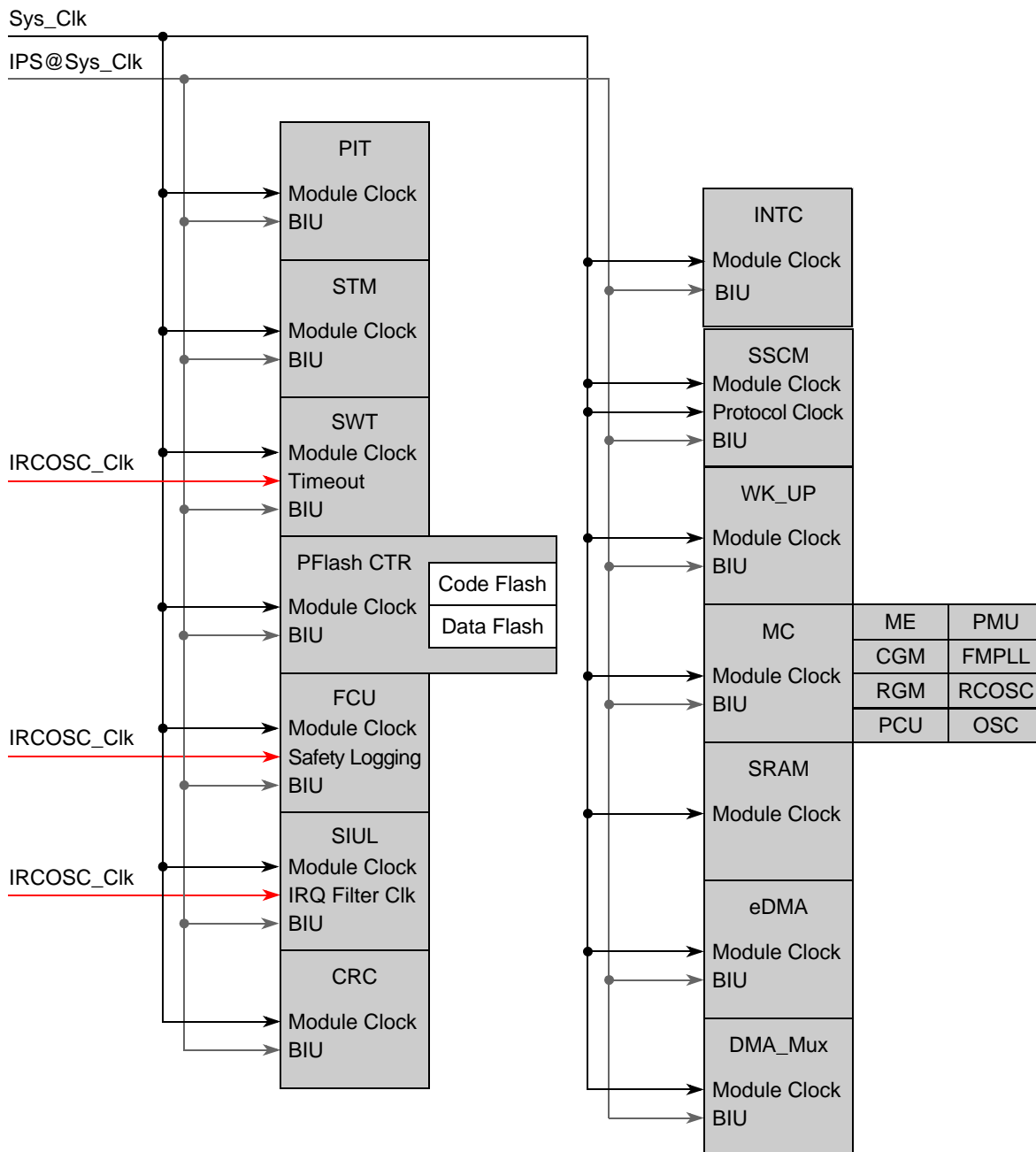


Figure 9. MPC5604E Clock Distribution Part A

Figure 10 shows how the clocks are connected to the remaining system components and to the functional IP modules. Again, all BIUs (bus interface units) are running on system clock that is gated on only during active IPS bus cycles.

Furthermore, the functional clocks RTC_Clk, Vid_Clk, and A0/1/2_Clk are routed to the corresponding IP modules. The RTC_Clk can be selected as time base for the precision time stamps of the PTP. The A0/1/2_Clk clocks provide the protocol clock for the audio interfaces. The Vid_Clk drives the logic of the video encoder and its input- and output buffers.

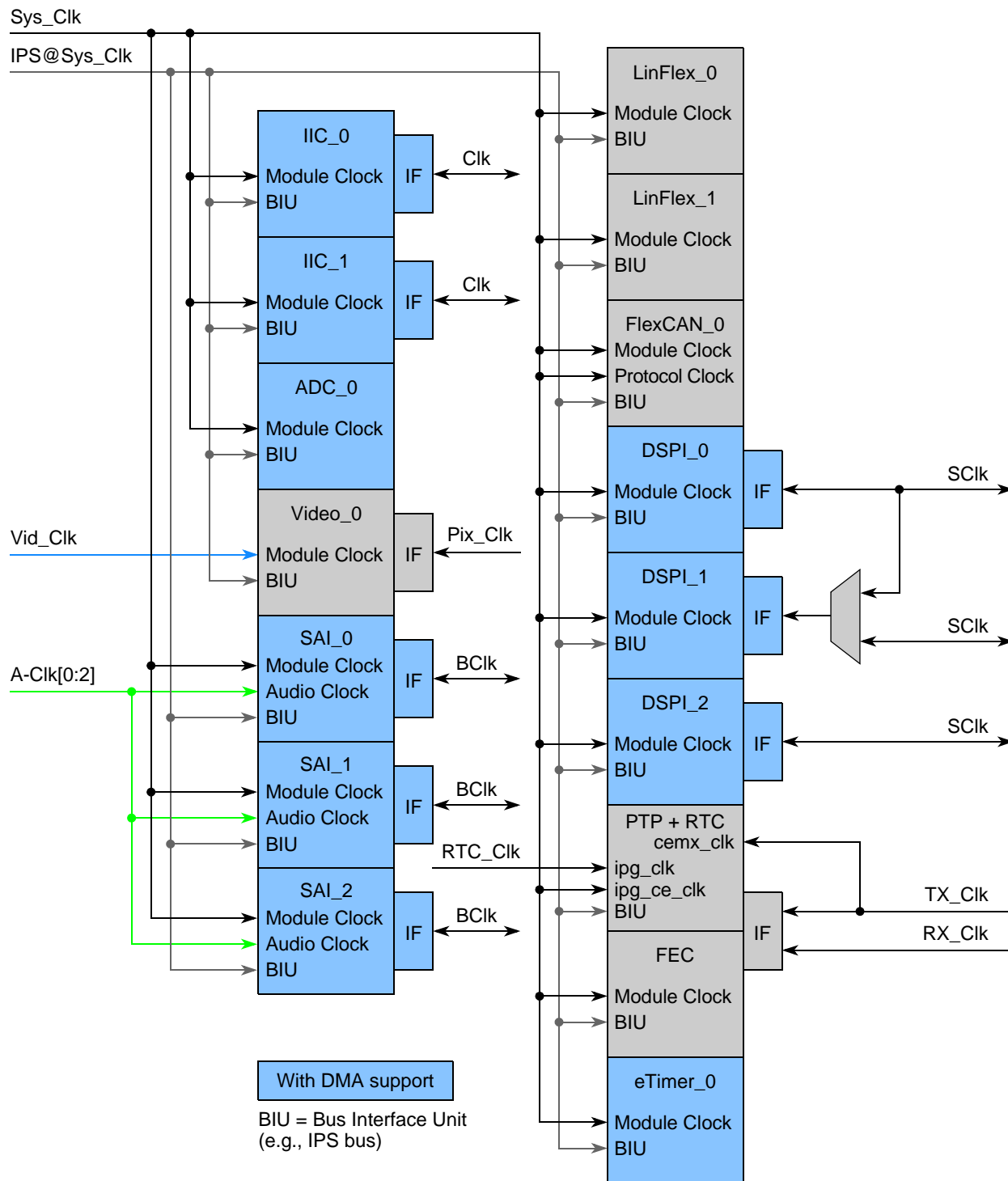


Figure 10. MPC5604E Clock Distribution Part B

NOTE

Disabling of peripherals and clock sources:

Software should take care that the clock sources (system clocks and protocol clocks) clock sources selected for a peripheral are not disabled during any mode transition preceding a mode transition to disable the peripheral.

For example: In FLEXCAN, when FXOSC is selected for the CAN Engine clock source (FLEXCAN_CTRL[CLK_SRC]), before disabling FLEXCAN peripheral clock, the user must ensure that FXOSC is enabled in the current mode prior to the target mode transition that will disable the FLEXCAN.

FXOSC is enabled in the current mode prior to the target mode transition that will disable the FLEXCAN.

Enabling of peripherals and clock sources:

Software should take care that the clock sources (system clocks and protocol clocks) are enabled for the target mode, before initiating the target mode transition to enable the peripheral.

4.3 Memory Map

This section provides the memory map of the registers that correspond to clock related modules.

Table 7. Memory Map

Address	Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0xC3FE_0000 ... 0xC3FE_001C	XOSC registers																
0xC3FE_0020 ... 0xC3FE_005C	reserved																
0xC3FE_0060 ... 0xC3FE_007C	IRC registers																

Table 7. Memory Map

Address	Name	0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																	
0xC3FE_0080 ... 0xC3FE_009C	reserved																																	
0xC3FE_00A0 ... 0xC3FE_00BC	FMPLL_0 registers																																	
0xC3FE_00C0 ... 0xC3FE_00FC	reserved																																	
0xC3FE_0100 ... 0xC3FE_011C	CMU0 registers																																	
0xC3FE_0120	PLL_CLK_DIV	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																																
0xC3FE_0124 ... 0xC3FE_013C	reserved																																	
0xC3FE_0140	SYSTEM_CLK_DIV	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		W																																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																																
0xC3FE_0144 ... 0xC3FE_015C	reserved																																	

Table 7. Memory Map

Address	Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0xC3FE_0160	RTC_CLK_DIV	R	0	0	0	0	0	0	0	0	CLK_DIV	0	0	0	0	0	0	0	0
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																	
0xC3FE_0164 ... 0xC3FE_036C	reserved																		
0xC3FE_0370	CGM_OC_EN ¹	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
		W																	
0xC3FE_0374	CGM_OCDS_SC ¹	R	0	0	SELDIV			SELCTL			0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																	
0xC3FE_0378	CGM_SC_SS ¹	R	0	0	0	0	SELSTAT			0	0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																	

¹ For details, refer to the chapter Clock Generation Module (MC_CGM).

4.4 Internal RC oscillator (IRC) digital interface

4.4.1 Introduction

The IRC digital interface controls the 16 MHz fast internal RC oscillator (IRC). It holds control and status registers accessible for application.

4.4.2 Functional description

The IRC provides a high frequency clock of 16 MHz. This clock can be used to accelerate the exit from reset and wakeup sequence from low power modes of the system. It is controlled by the MC_ME module

based on the current device mode. The clock source status is updated in ME_GS[S_RC]. Please refer to the MC_ME chapter for further details.

The IRC can be further divided by a configurable division factor in the range from 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by RC_CTL[RCDIV] bits.

The IRC output frequency can be trimmed by RC_CTL[IRCTRIM] bits. These bits can be programmed to modify internal capacitor/resistor values. After power on reset, the trimming bits are provided by the flash options. Only after a first write access to RC_CTL will the value specified by bits IRCTRIM control the trimming.

In this oscillator, two's complement trimming method is implemented. So the trimming code increases from -32 to 31. As the trimming code increases, the internal time constant increases and frequency reduces. Please refer to device datasheet for average frequency variation of the trimming step.

4.4.3 Register description

Address offset: 0x0000												Base Address: 0xC3FE_0060				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Reserved											IRCTRIM[5:0]					
Access											rw					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Reserved																
Access				rw				—				rw				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 11. IRC Oscillator Control Register (IRC_CTL)

Table 8. RC Oscillator Control Register (IRC_CTL) field descriptions

Field	Description
IRCTRIM[5:0]	IRC trimming bits Note: All configurations cannot be used. Please refer to data sheet.

NOTE

The IRC is trimmed during reset using a factory programmed value stored in flash. After reset, this trim value is not visible at IRC_CTL[TRIM]. Therefore, any read-write operation on the 32-bit register will potentially set the IRC to an unoptimised trim value.

Bus access errors are generated in only half of the non-implemented address space of Oscillator External Interface (Crystal XOSC) and RCOSC Digital Interface (16MHz Internal RC oscillator [IRC]). Do not access unimplemented address space for XOSC and RCOSC register areas OR write software that is not dependent on receiving an error when access to unimplemented XOSC and RCOSC space occurs.

4.5 External crystal oscillator (XOSC) digital interface

The XOSC digital interface controls the operation of the 4– MHz fast external crystal oscillator (XOSC). It holds control and status registers accessible for application.

4.5.1 Main features

- Oscillator powerdown control and status reporting through MC_ME block
- Oscillator clock available interrupt
- Oscillator bypass mode
- Output clock division factors ranging from 1, 2, 3....32

4.5.2 Functional description

The XOSC circuit includes an internal oscillator driver and an external crystal circuitry. It provides an output clock that can be provided to the FMPLL or used as a reference clock to specific modules depending on system needs.

The XOSC can be controlled by the MC_ME module. The ME_XXX_MC[XOSCON] bit controls the powerdown of the oscillator based on the current device mode while ME_GS[S_XOSC] register provides the oscillator clock available status.

After system reset, the oscillator is put into powerdown state and software has to switch on when required. Whenever the crystal oscillator is switched on from the off state, the OSCCNT counter starts and when it reaches the value $EOCV[7:0] \times 512$, the oscillator clock is made available to the system. Also, an interrupt pending XOSC_CTL[I_OSC] bit is set. An interrupt is generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This bit can only be set by software. A system reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as the external clock applied on the EXTAL pin and the oscillator status is forced to '1'. The bypass configuration is independent of the powerdown mode of the oscillator.

[Table 9](#) shows the truth table of different oscillator configurations.

Table 9. Truth table of crystal oscillator

ME_XXX_MC[FXOSCON]	FXOSC_CTL[OSCBYP]	XTAL	EXTAL	FXOSC	Oscillator MODE
0	0	No crystal, High Z	No crystal, High Z	0	Powerdown
x	1	x	Ext clock	EXTAL	Bypass, OSC disabled
1	0	Crystal	Crystal	EXTAL	Normal, OSC enabled
		Gnd	Ext clock	EXTAL	Normal, OSC enabled

The XOSC clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by XOSC_CTL[OSCDIV] field.

4.5.3 Register description

Address offset: 0x0000 Base Address: 0xC3FE0000

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OSCBYP	Reserved								EOCV[7:0]							
Access	rs				—								rw			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
M_OSC	Reserved								I_OSC	Reserved							
Access	rw		—		rw				rc	—							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 12. External Crystal Oscillator Control Register (XOSC_CTL)

Table 10. External Crystal Oscillator Control Register (XOSC_CTL) field descriptions

Field	Description
OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Only software can set this bit. System reset is needed to clear this bit. 0: Oscillator output is used as root clock 1: EXTAL is used as root clock
EOCV[7:0]	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state (OSCCNT runs on the FXOSC). This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV[7:0] × 512, the crystal oscillator clock interrupt (I_OSC) request is generated. The OSCCNT counter will be kept under reset if oscillator bypass mode is selected.
M_OSC	Crystal oscillator clock interrupt mask 0: Crystal oscillator clock interrupt is masked 1: Crystal oscillator clock interrupt is enabled
I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when OSCCNT counter reaches the count value EOCV[7:0]×512. It is cleared by software by writing '1'. 0: No oscillator clock interrupt occurred 1: Oscillator clock interrupt pending

NOTE

Bus access errors are generated in only half of the non-implemented address space of Oscillator External Interface (Crystal XOSC) and RCOSC Digital Interface (16MHz Internal RC oscillator [IRC]). Do not access unimplemented address space for XOSC and RCOSC register areas OR write software that is not dependent on receiving an error when access to unimplemented XOSC and RCOSC space occurs."

4.6 Frequency-modulated phase-locked loop (FMPLL)

4.6.1 Introduction

This section describes the features and functions of the FMPLL module implemented in the device.

4.6.2 Overview

The FMPLL enables the generation of high speed system clocks from a common 4–40 MHz input clock. Further, the FMPLL supports programmable frequency modulation of the system clock. The FMPLL multiplication factor and output clock divider ratio are all software configurable.

MPC5604E has one FMPLL that can generate the system clock and takes advantage of the FM mode.

NOTE

The user must take care not to program device with a frequency higher than allowed (no hardware check).

The FMPLL block diagram is shown in [Figure 13](#).

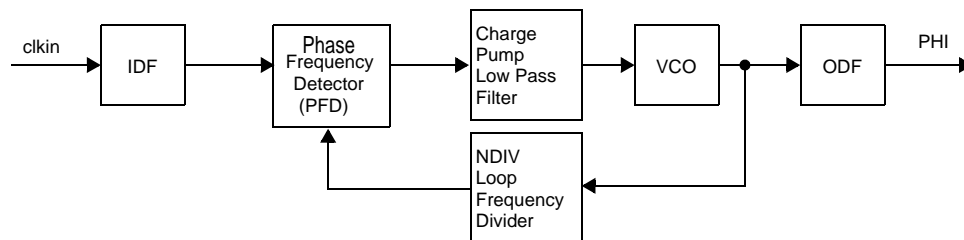


Figure 13. FMPLL block diagram

4.6.3 Features

The FMPLL has the following major features:

- Input clock frequency 4 MHz – 40 MHz
- PFD input clock frequency range in normal mode is 4–16 MHz
- Voltage controlled oscillator (VCO) range from 256 MHz to 512 MHz
- Frequency divider (FD) for reduced frequency operation without forcing the FMPLL to relock
- Frequency modulated FMPLL
 - Modulation enabled/disabled through software
 - Triangle wave modulation
- Programmable modulation depth
 - $\pm 0.25\%$ to $\pm 4\%$ deviation from center spread frequency¹
 - -0.5% to $+8\%$ deviation from down spread frequency
 - Programmable modulation frequency dependent on reference frequency
- Self-locked mode (SCM) operation
- 4 available modes
 - Normal mode
 - Progressive clock switching
 - Normal mode with frequency modulation
 - Powerdown mode

4.6.4 Memory map²

[Table 11](#) shows the memory map of the FMPLL. The FMPLL_0 base address is 0xC3FE_00A0.

1. Spread spectrum should be programmed in line with maximum datasheet frequency figures.
 2. FMPLL_x are mapped through the MC_CGM register slot

Table 11. FMPLL memory map

Base address: 0xC3FE00A0 (FMPLL_0)			
Address offset	Register	Access	Location
0x0	Control Register (CR)	R/W (write access in supervisor mode only)	on page 67
0x4	Modulation Register (MR)	R/W (write access in supervisor mode only)	on page 69

Table 12. FMPLL memory map

Address offset	Register	Location
0x0	Control Register (CR)	on page 67
0x4	Modulation Register (MR)	on page 69

4.6.5 Register description

The FMPLL operation is controlled by two registers. Those registers can be accessed and written in supervisor mode only.

4.6.5.1 Control Register (CR)

Table 13. CR field descriptions

Field	Description
IDF	The value of this field sets the FMPLL input division factor as described in Table 14 .
ODF	The value of this field sets the FMPLL output division factor as described in Table 15 .
NDIV	The value of this field sets the FMPLL loop division factor as described in Table 16 .
EN_PLL_SW	This bit is used to enable progressive clock switching. After the PLL locks, the PLL output initially is divided by 8, and then progressively decreases until it reaches divide-by-1. 0 Progressive clock switching disabled. 1 Progressive clock switching enabled. Note: Progressive clock switching should not be used if a non-changing clock is needed, such as for serial communications, until the division has finished.
I_LOCK	This bit is set by hardware whenever there is a lock/unlock event.
S_LOCK	This bit is an indication of whether the FMPLL has acquired lock. 0: FMPLL unlocked 1: FMPLL locked Note: S_LOCK =1 signals coarse lock. The system clock should not be changed to PLL output for at least 200 μ s after S_LOCK is set.

Table 13. CR field descriptions (continued)

Field	Description
PLL_FAIL_MASK	This bit is used to mask the pll_fail output. 0 pll_fail not masked. 1 pll_fail masked.
PLL_FAIL_FLAG	This bit is asynchronously set by hardware whenever a loss of lock event occurs while FMPLL is switched on. It is cleared by software writing '1'.

Table 14. Input divide ratios

IDF[3:0]	Input divide ratios
0000	Divide by 1
0001	Divide by 2
0010	Divide by 3
0011	Divide by 4
0100	Divide by 5
0101	Divide by 6
0110	Divide by 7
0111	Divide by 8
1000	Divide by 9
1001	Divide by 10
1010	Divide by 11
1011	Divide by 12
1100	Divide by 13
1101	Divide by 14
1110	Divide by 15
1111	Clock Inhibit

Table 15. Output divide ratios

ODF[1:0]	Output divide ratios
00	Divide by 2
01	Divide by 4
10	Divide by 8
11	Divide by 16

Table 16. Loop divide ratios

NDIV[6:0]	Loop divide ratios
0000000–0011111	—
0100000	Divide by 32
0100001	Divide by 33
0100010	Divide by 34
...	...
1011111	Divide by 95
1100000	Divide by 96
1100001–1111111	—

4.6.5.2 Modulation Register (MR)

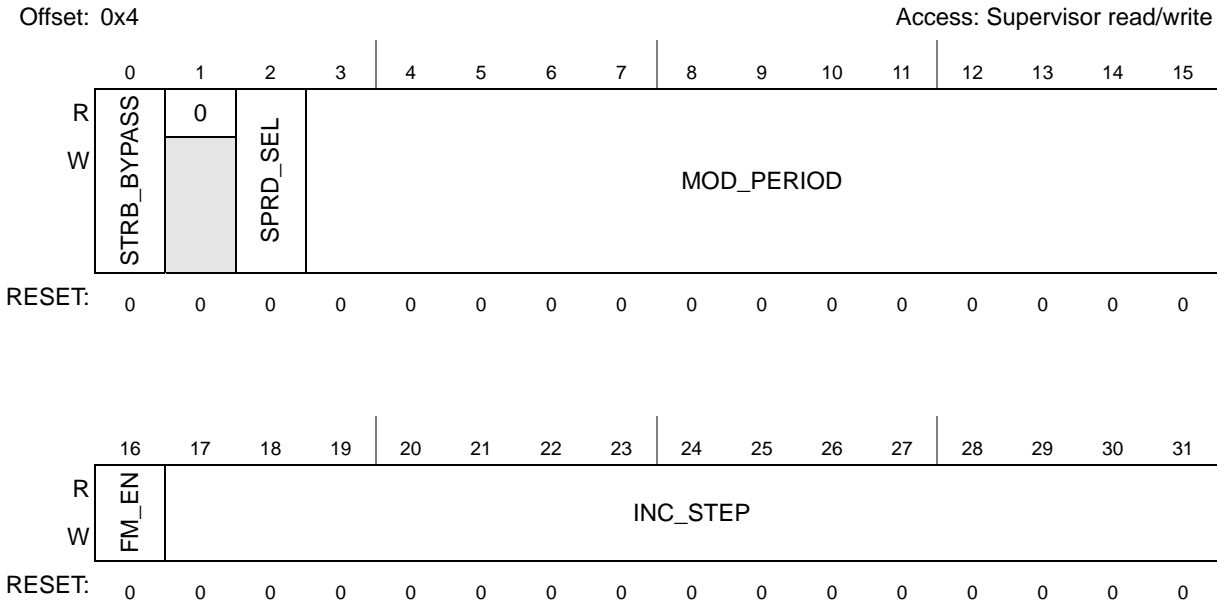


Figure 14. Modulation Register (MR)

Table 17. MR field descriptions

Field	Description
STRB_BYPASS	<p>Strobe bypass.</p> <p>The STRB_BYPASS signal is used to bypass the strobe signal used inside FMPLL to latch the correct values for control bits (INC_STEP, MOD_PERIOD and SPRD_SEL).</p> <p>0 Strobe is used to latch FMPLL modulation control bits</p> <p>1 Strobe is bypassed. In this case control bits need to be static. The control bits must be changed only when FMPLL is in powerdown mode.</p>
SPRD_SEL	<p>Spread type selection.</p> <p>The SPRD_SEL controls the spread type in Frequency Modulation mode.</p> <p>0 Center SPREAD</p> <p>1 Down SPREAD</p>
MOD_PERIOD	<p>Modulation period.</p> <p>The MOD_PERIOD field is the binary equivalent of the value modperiod derived from following formula:</p> $\text{modperiod} = \frac{f_{\text{ref}}}{4 \times f_{\text{mod}}}$ <p>where:</p> <p>f_{ref}: represents the frequency of the feedback divider</p> <p>f_{mod}: represents the modulation frequency</p>
FM_EN	<p>Frequency Modulation Enable. The FM_EN enables the frequency modulation.</p> <p>0 Frequency modulation disabled</p> <p>1 Frequency modulation enabled</p>
INC_STEP	<p>Increment step.</p> <p>The INC_STEP field is the binary equivalent of the value incstep derived from following formula:</p> $\text{incstep} = \text{round}\left(\frac{(2^{15} - 1) \times \text{md} \times \text{MDF}}{100 \times 5 \times \text{MODPERIOD}}\right)$ <p>where:</p> <p>md: represents the peak modulation depth in percentage (Center spread -- pk-pk=+/-md, Downspread -- pk-pk=-2xmd)</p> <p>MDF: represents the nominal value of loop divider (CR[NDIV])</p>

4.6.6 Functional description

4.6.6.1 Normal mode

In Normal Mode the FMPLL inputs are driven by the CR. This means that, when the FMPLL is in lock state, the FMPLL output clock (PHI) is derived by the reference clock (CLKIN) through this relation:

$$\text{phi} = \frac{\text{clk}_{\text{in}} \cdot \text{NDIV}}{\text{IDF} \cdot \text{ODF}}$$

where the value of IDF, NDIV and ODF are set in the CR and can be derived from [Table 14](#), [Table 15](#) and [Table 16](#).

4.6.6.2 Progressive clock switching

Progressive clock switching allows to switch the system clock to FMPLL output clock stepping through different division factors. This means that the current consumption gradually increases and, in turn, voltage regulator response is improved.

This feature can be enabled by programming CR[EN_PLL_SW] bit. When enabled, the system clock is switched to divided PHI. The FMPLL_clk divider is then progressively decreased to the target divider as shown in [Table 18](#).

Table 18. Progressive clock switching on pll_select rising edge

Number of FMPLL output clock cycles	FMPLL_clk frequency (FMPLL output clock frequency)
8	(FMPLL output clock frequency)/8
16	(FMPLL output clock frequency)/4
32	(FMPLL output clock frequency)/2
onward	FMPLL output clock frequency

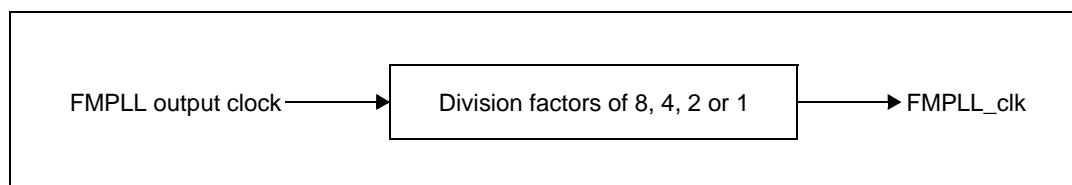


Figure 15. FMPLL output clock division flow during progressive switching

4.6.6.3 Normal mode with frequency modulation

The FMPLL default mode is without frequency modulation enabled. When frequency modulation is enabled, however, two parameters must be set to generate the desired level of modulation: the PERIOD, and the STEP. The modulation waveform is always a triangle wave and its shape is not programmable.

FM mode is activated in two steps:

1. Configure the FM mode characteristics: MOD_PERIOD, INC_STEP.
2. Enable the FM mode by programming bit FM_EN of the MR to '1'. FM mode can only be enabled when FMPLL is in lock state.

There are two ways to latch these values inside the FMPLL, depending on the value of bit STRB_BYPASS in the MR.

If STRB_BYPASS is low, the modulation parameters are latched in the FMPLL only when the strobe signal goes high for at least two cycles of CLKIN clock. The strobe signal is automatically generated in the FMPLL digital interface when the modulation is enabled (FM_EN goes high) if the FMPLL is locked

(S_LOCK = 1) or when the modulation has been enabled (FM_EN = 1) and FMPLL enters lock state (S_LOCK goes high).

If STRB_BYPASS is high, the strobe signal is bypassed. In this case, control bits (MOD_PERIOD[12:0], INC_STEP[14:0], SPREAD_CONTROL) need to be static or hardwired to constant values. The control bits must be changed only when the FMPLL is in powerdown mode.

The modulation depth in % is

$$\text{ModulationDepth} = \left(\frac{100 \times 5 \times \text{INCSTEP} \times \text{MODPERIOD}}{(2^{15} - 1) \times \text{MDF}} \right)$$

NOTE

The user must ensure that the product of INCSTEP and MODPERIOD is less than $(2^{15}-1)$.

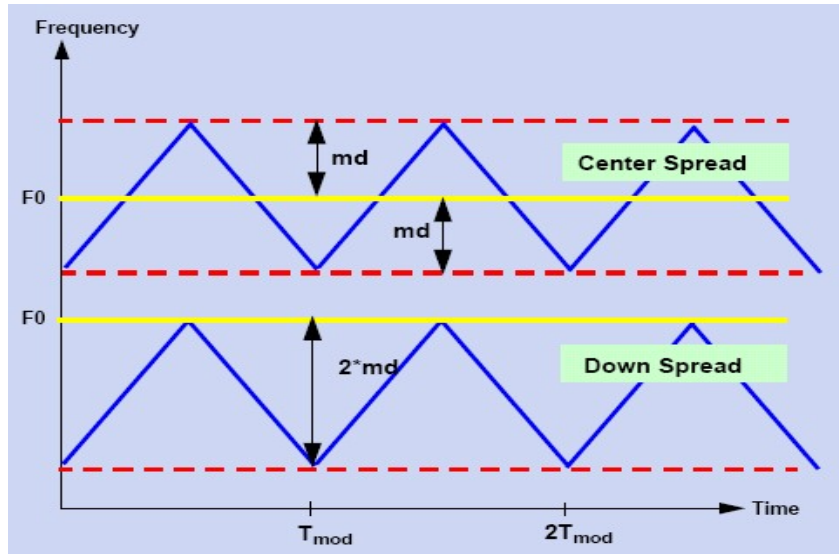


Figure 16. Frequency modulation

4.6.6.4 Powerdown mode

To reduce consumption, the FMPLL can be switched off when not required by programming the registers ME_x_MC on the MC_ME module.

4.6.7 Recommendations

To avoid any unpredictable behavior of the FMPLL clock, it is recommended to follow these guidelines:

- The FMPLL VCO frequency should reside in the range 256 MHz to 512 MHz. Care is required when programming the multiplication and division factors to respect this requirement.
- The user must change the multiplication, division factors only when the FMPLL output clock is not selected as system clock. Use progressive clock switching if system clock changes are required

while the PLL is being used as the system clock source. MOD_PERIOD, INC_STEP, SPREAD_SEL bits should be modified before activating the FM mode. Then strobe has to be generated to enable the new settings. If STRB_BYP is set to '1' then MOD_PERIOD, INC_STEP and SPREAD_SEL can be modified only when FMPLL is in powerdown mode.

- FMPLL must be powered down before changing the values of NDIV and IDF and then powered up with the new settings. If the power up and power down is not performed, the NDIV and IDF will reflect new configuration values but the PLL frequency will be as per the old configuration.
- Use progressive clock switching (FMPLL output clock can be changed when it is the system clock, but only when using progressive clock switching).
- Before enabling FMPLL, ensure that the input clock to FMPLL is stable.

4.7 Clock Monitor Unit (CMU)

4.7.1 Overview

The Clock Monitor Unit (CMU) serves three purposes:

- FMPLL clock monitoring: detect if FMPLL leaves an upper or lower frequency boundary
- Crystal clock monitoring: monitor the external crystal oscillator clock, which must be greater than the internal RC clock divided by a division factor given by CMU_CSR[RCDIV]
- Frequency meter: measure the frequency of RCOSC versus a known reference clock XOSC

CMU forwards these kind of events to the MC_CGM, MC_ME, and FCU. These in turn can then switch to a safe mode, generate a reset, or generate an interrupt.

Table 19. CMU module summary

Module	Monitored clocks
CMU_0	PLL divider clock

4.7.2 Main features

- RC oscillator frequency measurement
- External oscillator clock monitoring with respect to CK_IRC÷n clock
- FMPLL clock frequency monitoring with respect to CK_IRC÷4 clock
- Event generation for various failures detected inside monitoring unit

4.7.3 Functional description

The names of the clocks involved in this block have the following meaning:

- CK_XOSC: clock coming from the external crystal oscillator
- CK_IRC: clock coming from the low frequency internal RC oscillator
- CK_PLL: clock coming from the FMPLL
- FOSC: frequency of external crystal oscillator clock

- FRC: frequency of low frequency internal RC oscillator
- FPLL: frequency of FMPLL divider clock

4.7.4 Crystal clock monitor

If FOSC is smaller than FRC divided by 2^{RCDIV} bits of CMU_0_CSR and the CK_XOSC is 'ON' and stable as signaled by the MC_ME, then:

- CMU_ISR[OLRI] is set.

NOTE

OSC and FMPLL monitor may produce false events when OSC/FMPLL frequency is less than $2 \times \text{RC} / 2^{\text{RCDIV}}$ frequency due to the limitation on to compare two clocks accurately.

4.7.4.1 FMPLL clock monitor

The FMPLL clock CK_PLL frequency can be monitored by programming CME bit of CMU_CSR register to 1. CK_PLL monitor starts as soon as CME bit is set. This monitor can be disabled at any time by writing CME bit to 0.

If CK_PLL frequency (FPLL) is greater than a reference value determined by the HFREF[11:0] bits of CMU_HFREFR and the CK_PLL is 'ON' and the FMPLL locked as signaled by the MC_ME then

- An event pending bit FHHI in CMU_ISR is set

If FPLL is less than a reference clock frequency (FRC/4) and the CK_PLL is 'ON' and the FMPLL locked as signaled by the MC_ME, then:

- An event pending bit FLCI in CMU_ISR is set

If FPLL is less than a reference value determined by the LFREF[11:0] bits of CMU_LFREFR and the CK_PLL is 'ON' and the FMPLL locked as signaled by the MC_ME, then:

- An event pending bit FLLI in CMU_ISR is set
- A failure event is signaled to the MC_RGM and Fault Collection Unit, which in turn can generate an interrupt

NOTE

OSC and FMPLL monitor may produce false events when OSC/FMPLL frequency is less than $2 \times \text{RC} / 2^{\text{RCDIV}}$ frequency due to the limitation on to compare two clocks accurately.

4.7.4.2 Frequency meter

The frequency meter calibrates the internal RC oscillator (CK_IRC) using a known frequency.

NOTE

This value can then be stored in the flash memory so that application software can reuse it later on.

The reference clock is always the XOSC. A simple frequency meter returns a draft value of CK_IRC. The measure starts when CMU_CSR[SFM] is set. The measurement duration is given by the CMU_MDR register in terms of IRC clock cycles with a width of 20 bits. The SFM bit is cleared by the hardware after the frequency measurement is done and the count is loaded in the CMU_FDR. The frequency FRC can be derived from the value loaded in the CMU_FDR register as follows:

$$\text{FRC} = (\text{FOSC} \times \text{MD}) / n \quad \text{Eqn. 1}$$

where n is the value in CMU_FDR register and MD is the value in CMU_MDR.

NOTE

When $\text{FXOSC} > \text{FIRC}$ and $\text{MDR} = 0\text{x}\text{FFFFFF}$, it will cause overflow of FDR register and the value stored cannot be relied upon. When $\text{FXOSC} > \text{FIRC}$, application need to ensure that value of MDR is chosen such that n does not overflow where n is number of XOSC clock edges in duration of MDR IRC clock edges.

4.7.5 Memory map and register description

The CMU registers are mapped through the MC_CGM (see the memory map in [Chapter 5, Clock Generation Module \(MC_CGM\)](#)). The base address for CMU is shown in [Table 20](#).

Table 20. CMU base address

Module	Base address
CMU_0	0xC3FE_0100

The memory map of CMU is shown in [Table 21](#).

Table 21. CMU memory map

Address offset	Register	Location
0x00	Control status register (CMU_CSR)	on page 76
0x04	Frequency display register (CMU_FDISP)	on page 77
0x08	High-frequency reference register A (CMU_HFREFR_A)	on page 77
0x0C	Low-frequency reference register A (CMU_LFREFR_A)	on page 78
0x10	Interrupt status register (CMU_ISR)	on page 78
0x14	Reserved	
0x18	Measurement duration register (CMU_MDR)	on page 79

4.7.5.1 Control status register (CMU_CSR)

Address: Base + 0x00

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SFM	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CKSEL1		0	0	0	0	0	RCDIV		CME_A
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 17. Control status register (CMU_CSR)

Table 22. CMU_0_CSR field descriptions

Field	Description
SFM	Start frequency measure Set this bit to start a clock frequency measure. The bit is cleared by hardware when the measure is ready in the CMU_FDR register. Software cannot clear this bit. 0 Frequency measurement is completed or not yet started. 1 Frequency measurement is started.
CKSEL1	Clock selection This field selects the clock to be measured by the frequency meter. Selects IRC 16 MHz as reference clock. This field must not be programmed with value 2.
RCDIV	RC clock division factor These bits specify the RC clock division factor. The output clock is CK_IRC fast (16 MHz) divided by the factor 2^{RCDIV} . This output clock is compared with CK_XOSC for crystal clock monitor feature. The clock division coding is as follows. 00 Clock divided by 1 (No division). 01 Clock divided by 2. 10 Clock divided by 4. 11 Clock divided by 8.
CME_A	Clock monitor enable 0 Monitor is disabled. 1 Monitor is enabled.

4.7.5.2 Frequency display register (CMU_FDR)

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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FD[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18. Frequency display Register (CMU_FDR)

Table 23. CMU_FDR field descriptions

Field	Description
FD	Measured frequency bits This register displays the measured frequency f_{RC} with respect to f_{OSC} . The measured value is given by the following formula: $f_{RC} = (f_{OSC} \times MD) / n$ where n is the value in CMU_FDR register

4.7.5.3 High-frequency reference register A (CMU_HFREFR_A)

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

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

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

Figure 19. High-frequency reference register A (CMU_HFREFR_A)

Table 24. CMU_HFREFR_A field descriptions

Field	Description
HFREF_A	High-frequency reference value These bits determine the high reference value for the FMPLL clock. The reference value is given by: $(HFREF_A/16) \times (f_{RC}/4)$.

4.7.5.4 Low-frequency reference register A (CMU_LFREFR_A)

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

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

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

Figure 20. Low-frequency reference register A (CMU_LFREFR_A)

Table 25. CMU_LFREFR_A fields descriptions

Field	Description
LFREF_A	Low-frequency reference value These bits determine the low reference value for the FMPLL clock. The reference value is given by: $(LFREF_A/16) \times (f_{RC}/4)$.

4.7.5.5 Interrupt status register (CMU_ISR)

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

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FLCI_A	FHHL_A	FLLI_A	OLRI
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21. Interrupt status register (CMU_ISR)

Table 26. CMU_ISR field descriptions

Field	Description
FLCI_A	FMPLL clock frequency less than reference clock interrupt This bit is set by hardware when both of the following are true: <ul style="list-style-type: none"> • The FMPLL frequency becomes lower than reference clock frequency (FRC/4) value • CK_FMPLL is 'ON' and the FMPLL locked as signaled by the MC_ME. It can be cleared by software by writing 1. 0 No FLC event. 1 FLC event is pending.

Table 26. CMU_ISR field descriptions (continued)

FHHI_A	<p>FMPLL_0 Clock frequency higher than high reference interrupt This bit is set by hardware when both of the following are true:</p> <ul style="list-style-type: none"> • The FMPLL frequency becomes higher than HFREF_A value • CK_FMPLL is 'ON' and the FMPLL locked as signaled by the MC_ME. <p>It can be cleared by software by writing 1. 0 No FHH event. 1 FHH event is pending.</p>
FLLI_A	<p>FMPLL_0 Clock frequency less than low reference event This bit is set by hardware when both of the following are true:</p> <ul style="list-style-type: none"> • The FMPLL frequency becomes lower than LFREF_A value • CK_FMPLL is 'ON' and the FMPLL locked as signaled by the MC_ME. <p>It can be cleared by software by writing 1. 0 No FLL event. 1 FLL event is pending.</p>
OLRI	<p>Oscillator frequency less than RC frequency event This bit is set by hardware when both of the following are true:</p> <ul style="list-style-type: none"> • The frequency of CK_XOSC is less than CK_IRCfast/2^{RCDIV} frequency • CK_XOSC is 'ON' and stable as signaled by the MC_ME. <p>It can be cleared by software by writing 1. 0 No OLR event. 1 OLR event is pending.</p>

4.7.5.6 Measurement duration register (CMU_MDR)

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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MD[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22. Measurement duration register (CMU_MDR)

Table 27. CMU_MDR field descriptions

Field	Description
MD	<p>Measurement duration bits This register displays the measured duration in term of IRC clock cycles. This value is loaded in the frequency meter down-counter. When CMU_CSR[SFM] = 1, the down-counter starts counting.</p>

4.8 Boot and power management concept

At boot time, the MPC5604E is using the IRCOSC_Clk to start. At this time (after reset), the system clock divider is set to a value of "1", thus, the system clock is 16MHz. Then, the MPC5604E can configure the module to connect the external oscillator (XOSC) and the PLL. Once the PLL is locked, the SW can

change the system clock divider to "2" and switch the system clock selector to be driven by the PLL. One output of the PLL can be reconfigured without producing glitches on the clock signal. This clock is used to drive the system clock (Sys_Clk).

The other output clock of the PLL is used to drive all the other clocks. Because this port cannot be reconfigured free of glitches, all attached IP modules need to be gated off before changing the clock frequency.

The other clock selectors (Clk Out Sel and the FCDs) do not allow glitch-less clock selection. Thus, SW need to gate off all attached IP modules before changing the clock source.

In case the PLL lock gets lost during operation, the MPC5604E switches to safe mode and automatically selects the IRCOSC_Clk as input at the system clock selector so that the system clock runs now at 8MHz. In safe mode, the SW can switch the divider value again to "1" thus, the system clock is 16MHz. The loss of lock event also creates an entry in the Fault Collection Unit (FCU).

The MPC5604E also supports two low power modes, the STOP mode and the HALT mode. These modes distinguish between different clock and clock gating settings. In both modes the clocks need to be configured to allow a resume to the RUN mode. [Figure 23](#) shows which modules need to use the reliable IRCOSC_Clk that remains active during STOP mode. These are the safety relevant functions in

- FCU
- SWT

and the functions need for wake-up, which are

- digital input filters for external interrupt detection (part of the SIUL).

The non maskable interrupt (NMI) is driven via an input pin. This signal is routed purely in a combinatorial way through the system. This allows to wake-up as long as only the Mode Entry (ME) unit is clocked.

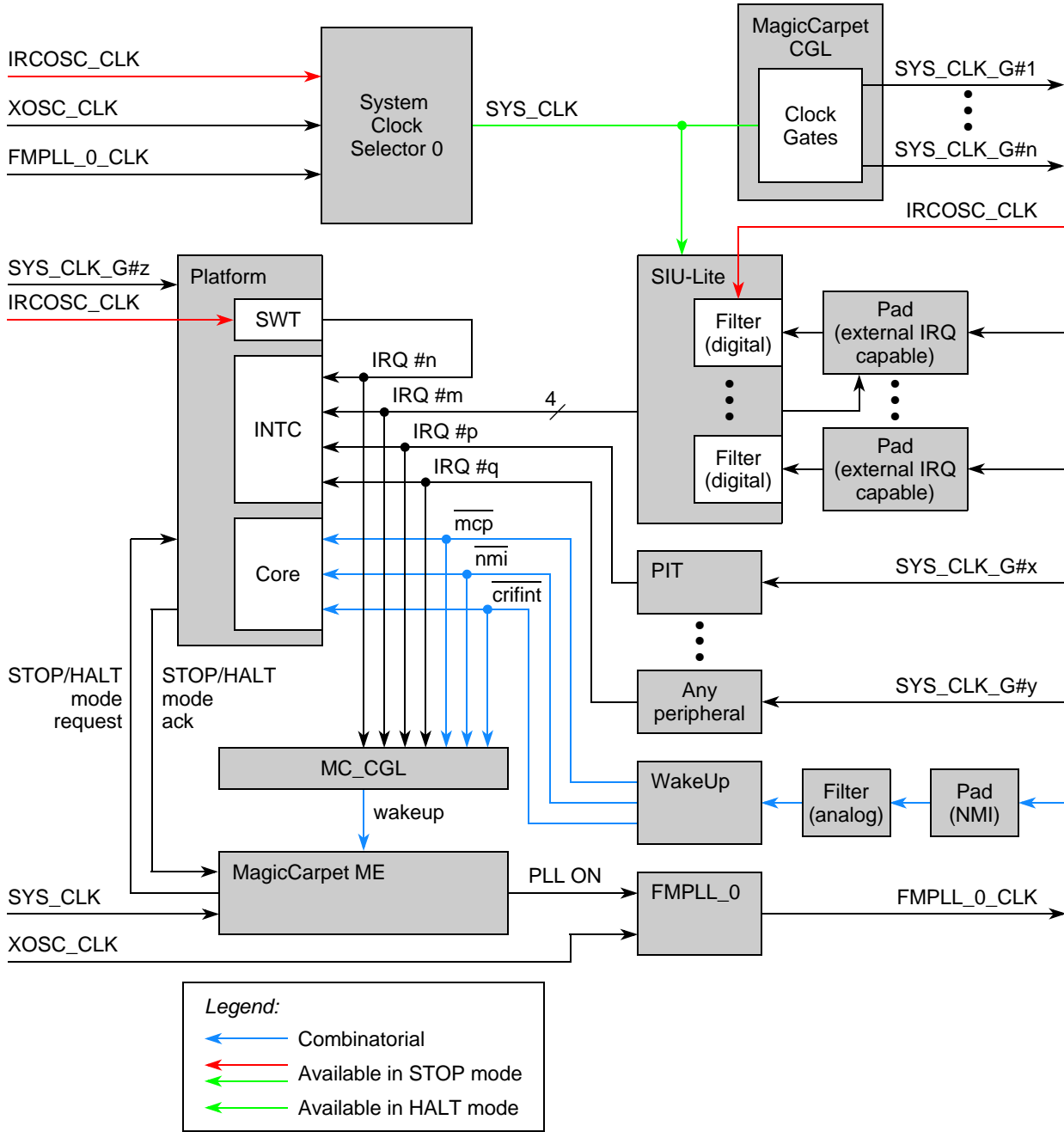


Figure 23. MPC5604E Clock Selection for Power Modes

4.9 Safety concept

This section gives a brief overview on the safety aspects of the clock architecture. This includes two items:

- reliable clock for safety modules
- clock frequency monitoring

A safety relevant unit of the MPC5604E is the Software Watchdog Timer (SWT). To ensure that this modules operates also in case of failure of the system clock, the watchdog counter decrements every cycle of the IRCOSC_Clk. This clock is more robust against failure compared to the PLL clock because it does not depend on other components. In case of the PLL clock, proper performance also depends on the external crystal or clock generator, the XOSC and loss of clock lock.

Besides the loss of PLL lock detection, the MPC5604E also has a Clock Monitoring Unit (CMU). Its main features are:

- RC oscillator (IRCOSC_Clk) frequency measurement
- External oscillator clock monitoring with respect to IRCOSC_Clk/n clock (turned off at reset)
- PLL clock frequency monitoring with respect to IRCOSC_Clk/4 clock (turned off at reset)
- Event generation for various failures detected inside monitoring unit

Figure 24 shows how the clock monitoring unit is integrated into the MPC5604E clocking system.

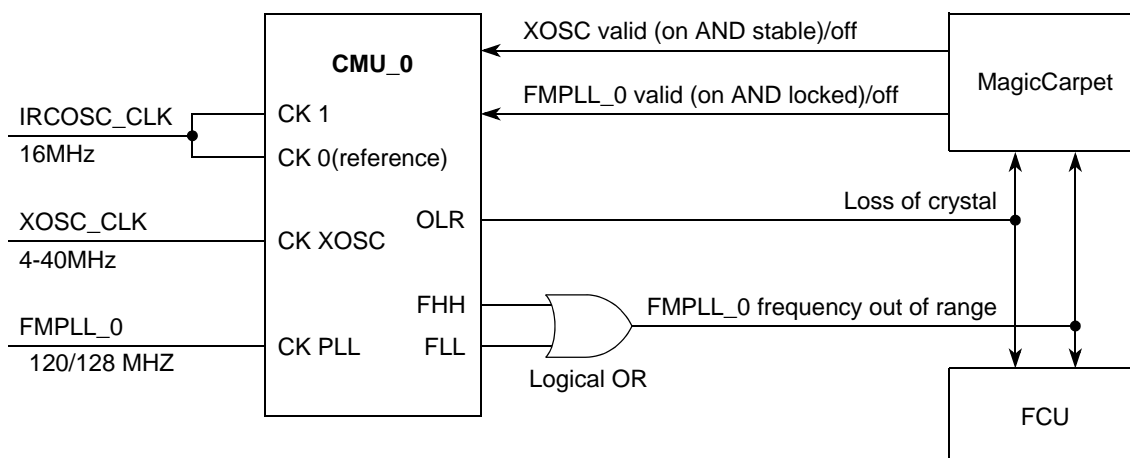


Figure 24. MPC5604E Clock Monitoring Unit Integration

In case of clock monitoring event, the CMU signals the Clock Generation Module, Reset Generation Module, and Mode Entry Module, which might take appropriate actions (e.g., switching the system clock to the IRCOSC_Clk), and to the fault collection unit (FCU) to keep record of the event.

Chapter 5

Clock Generation Module (MC_CGM)

5.1 Introduction

5.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space. The MC_CGM also selects and generates an output clock.

[Figure 25](#) shows the MC_CGM block diagram.

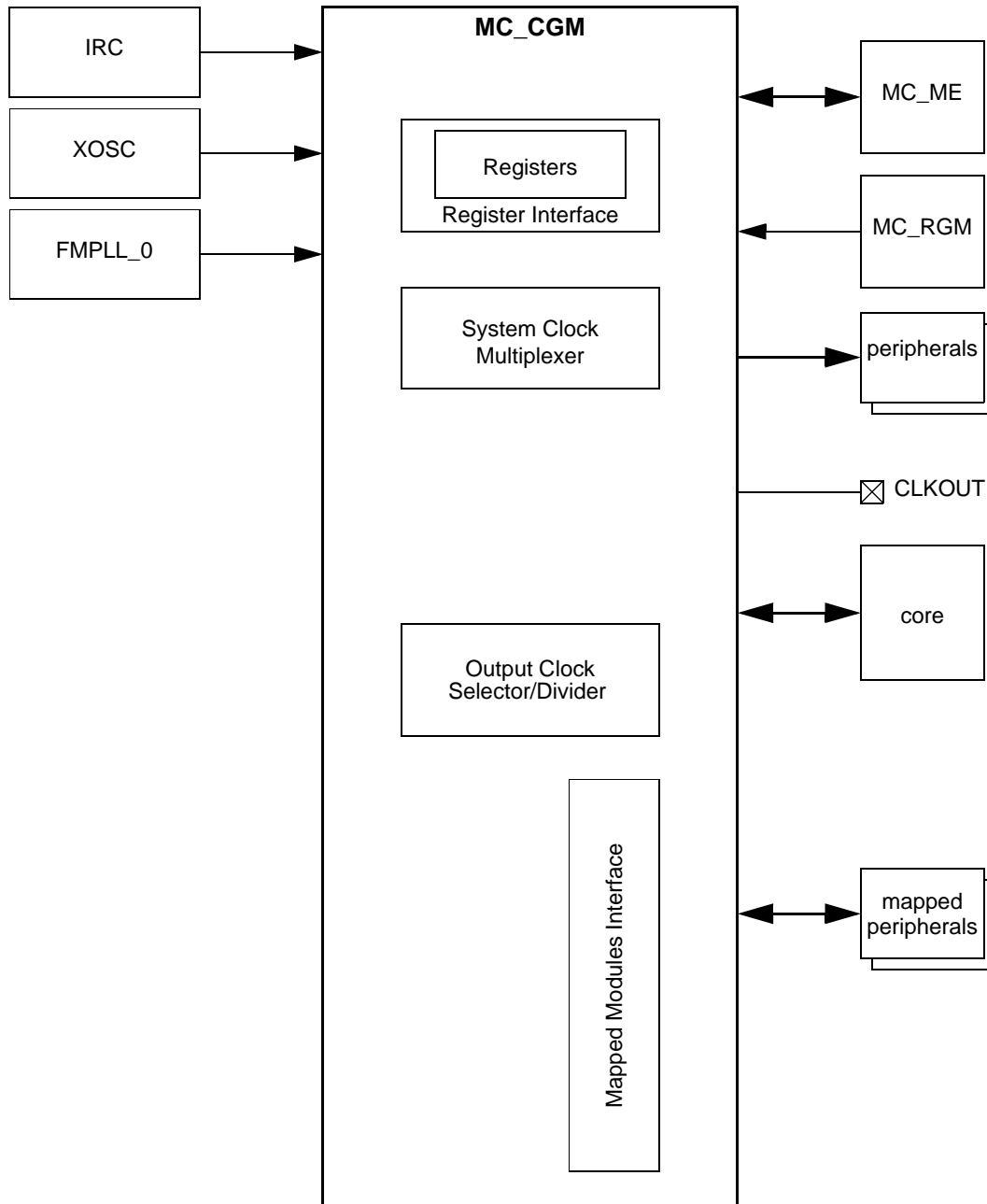


Figure 25. MC_CGM block diagram

5.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control

- supports multiple clock sources and maps their address spaces to its memory map
- generates an output clock
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

5.2 External Signal Description

The MC_CGM delivers an output clock to the CLKOUT pin for off-chip use and/or observation.

5.3 Memory Map and Register Definition

Table 28. MC_CGM Register Description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FE_0120	PLL_DIVIDER	PLL Clock Divider	byte	read	read/write	read/write	on page 86
0xC3FE_0140	SYSTEM_CLK_DIVIDER	System Clock Divider	byte	read	read/write	read/write	on page 86
0xC3FE_0160	RTC_CLK_DIVIDER	RTC Clock Divider	byte	read	read/write	read/write	on page 87
0xC3FE_0370	CGM_OC_EN	Output Clock Enable	word	read	read/write	read/write	on page 87
0xC3FE_0374	CGM_OCDS_SC	Output Clock Division Select	byte	read	read/write	read/write	on page 88
0xC3FE_0378	CGM_SC_SS	System Clock Select Status	byte	read	read	read	on page 89

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

5.3.1 Register Descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **CGM_OC_EN** register may be accessed as a word at address 0xC3FE_0370, as a half-word at address 0xC3FE_0372, or as a byte at address 0xC3FE_0373.

5.3.1.1 PLL Clock Divider Register (PLL_CLK_DIV)

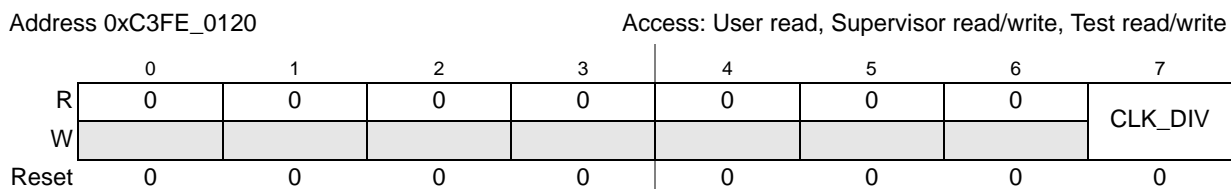


Figure 26. PLL Clock Divider Register (PLL_CLK_DIV)

This register sets the FMPLL_0 PCS clock division factor.

Table 29. PLL Clock Divider Register (PLL_CLK_DIV) Field Descriptions

Field	Description
CLK_DIV	Clock Divider Enable 0 FMPLL_0 PCS clock to system clock selector is divided by 1 (no division) 1 FMPLL_0 PCS clock to system clock selector is divided by 2

5.3.1.2 System Clock Divider Register (SYSTEM_CLK_DIV)

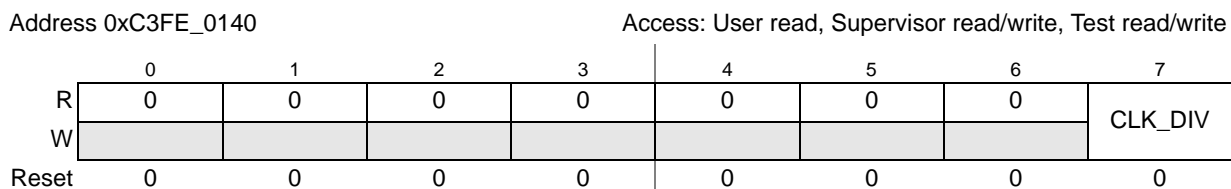


Figure 27. System Clock Divider Register (SYSTEM_CLK_DIV)

This register sets the system clock division factor.

Table 30. System Clock Divider Register (SYSTEM_CLK_DIV) Field Descriptions

Field	Description
CLK_DIV	Clock Divider Enable 0 system clock is divided by 1 (no division) 1 system clock is divided by 2

5.3.1.3 RTC Clock Divider Register (RTC_CLK_DIV)

Address 0xC3FE_0160 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	CLK_DIV
W								
Reset	0	0	0	0	0	0	0	0

Figure 28. RTC Clock Divider Register (RTC_CLK_DIV)

This register sets the RTC clock division factor.

Table 31. RTC Clock Divider Register (RTC_CLK_DIV) Field Descriptions

Field	Description
CLK_DIV	Clock Divider Enable 0 RTC clock is divided by 1 (no division) 1 RTC clock is divided by 2

5.3.1.4 Output Clock Enable Register (CGM_OC_EN)

Address 0xC3FE_0370 Access: User read, Supervisor read/write, Test read/write

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

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

Figure 29. Output Clock Enable Register (CGM_OC_EN)

This register is used to enable and disable the output clock.

Table 32. Output Clock Enable Register (CGM_OC_EN) Field Descriptions

Field	Description
EN	Output Clock Enable control 0 Output Clock is disabled 1 Output Clock is enabled

5.3.1.5 Output Clock Division Select Register (CGM_OCDS_SC)

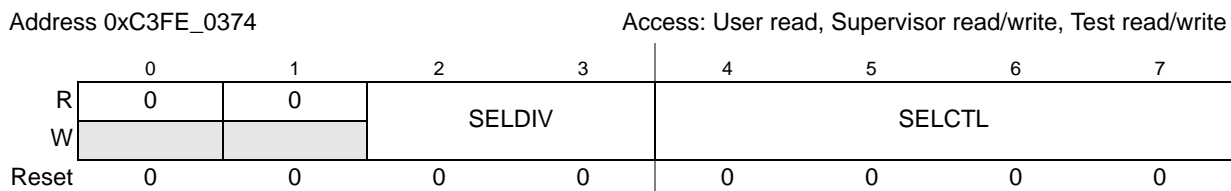


Figure 30. Output Clock Division Select Register (CGM_OCDS_SC)

This register is used to select the current output clock source and by which factor it is divided before being delivered at the output clock.

Table 33. Output Clock Enable Register (CGM_OC_EN) Field Descriptions

Field	Description
SELDIV	Output Clock Division Select 00 output selected Output Clock without division 01 output selected Output Clock divided by 2 10 output selected Output Clock divided by 4 11 output selected Output Clock divided by 8
SELCTL	Output Clock Source Selection Control — This value selects the current source for the output clock. 0000 IRC 0001 XOSC 0010 FMPLL_0 0011 FCD0 A0_CLK 0100 FCD1 A1_CLK 0101 FCD2 A2_CLK 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

5.3.1.6 System Clock Select Status Register (CGM_SC_SS)

Address 0xC3FE_0378 Access: User read, Supervisor read, Test read

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

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

Figure 31. System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Table 34. System Clock Select Status Register (CGM_SC_SS) Field Descriptions

Field	Description
SELSTAT	System Clock Source Selection Status — This value indicates the current source for the system clock. 0000 IRC 0001 reserved 0010 XOSC 0011 reserved 0100 FMPLL_0 PCS 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

5.4 Functional Description

5.4.1 System Clock Generation

Figure 32 shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the IRC as the system clock and to ignore the system clock select.

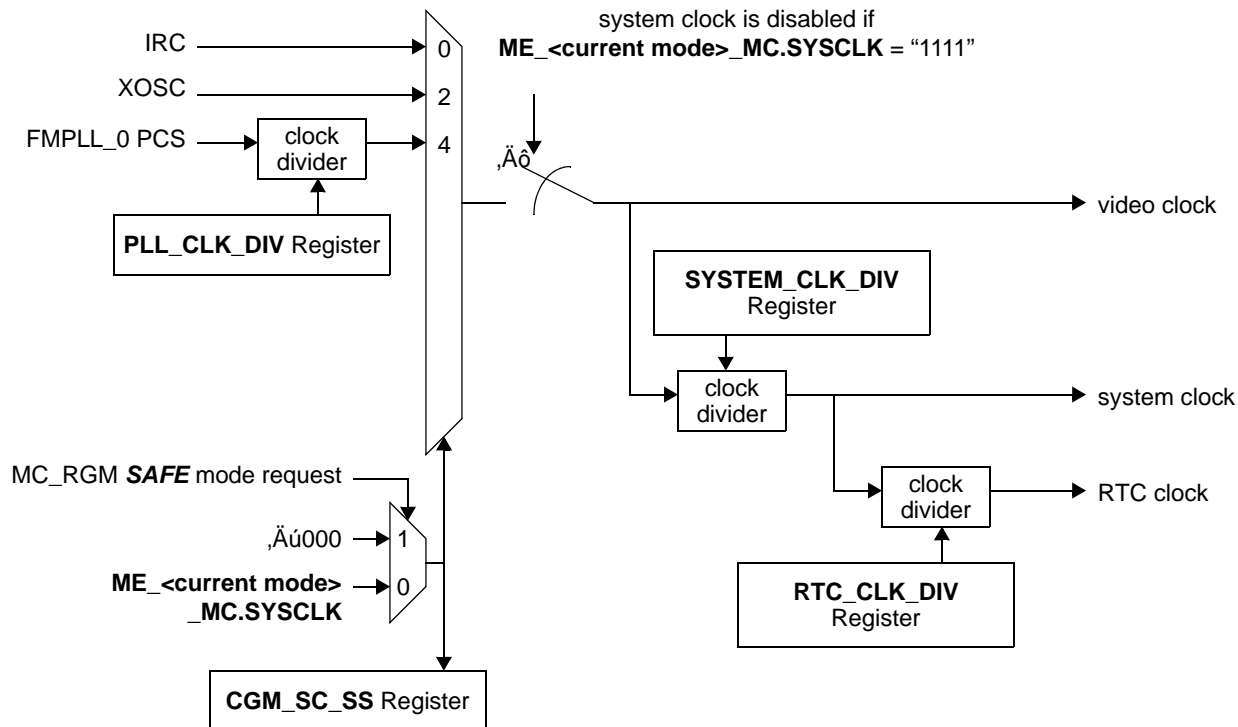


Figure 32. MC_CGM System Clock Generation Overview

5.4.1.1 System Clock Source Selection

During normal operation, the system clock selection is controlled

- on a **SAFE** mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

5.4.1.2 System Clock Disable

During the **TEST** mode, the system clock can be disabled by the MC_ME.

5.4.2 Output Clock Multiplexing

The MC_CGM contains a multiplexing function for a number of clock sources which can then be used as output clock sources. The selection is done via the **CGM_OCDS_SC** register.

5.4.3 Output Clock Division Selection

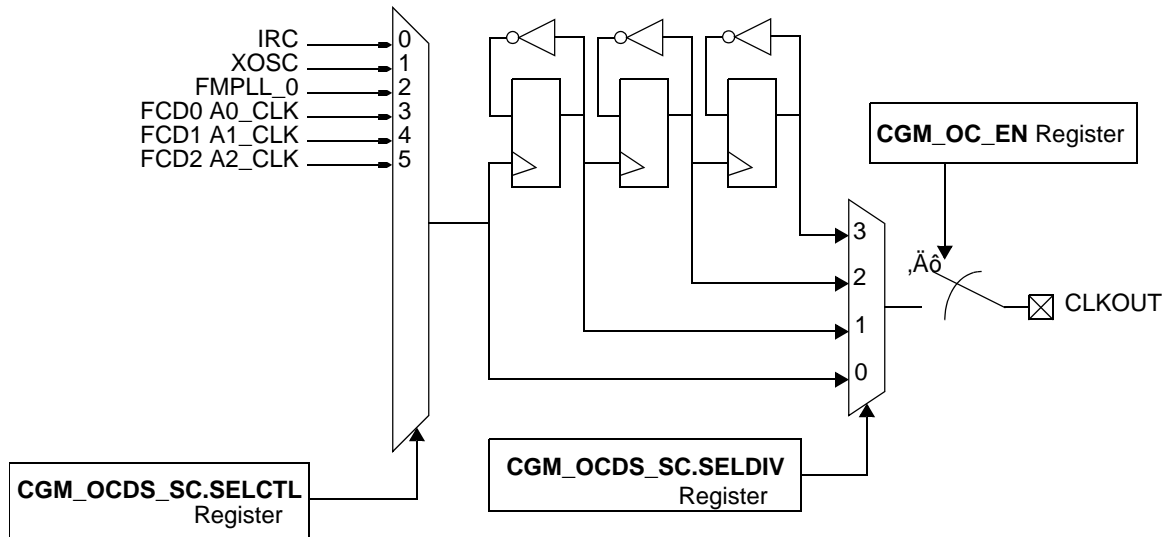


Figure 33. MC_CGM Output Clock Multiplexer and CLKOUT Generation

The MC_CGM provides the following output signal for the output clock generation:

- CLKOUT (see Figure 33). This signal is generated by using one of the 3-stage ripple counter outputs or the selected signal without division. The non-divided signal is not guaranteed to be 50% duty cycle by the MC_CGM.

The MC_CGM also has an output clock enable register (see Section 5.3.1.4, “Output Clock Enable Register (CGM_OC_EN)”) that contains the output clock enable/disable control bit.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 6

Mode Entry Module (MC_ME)

6.1 Introduction

6.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

[Figure 34](#) shows the MC_ME block diagram.

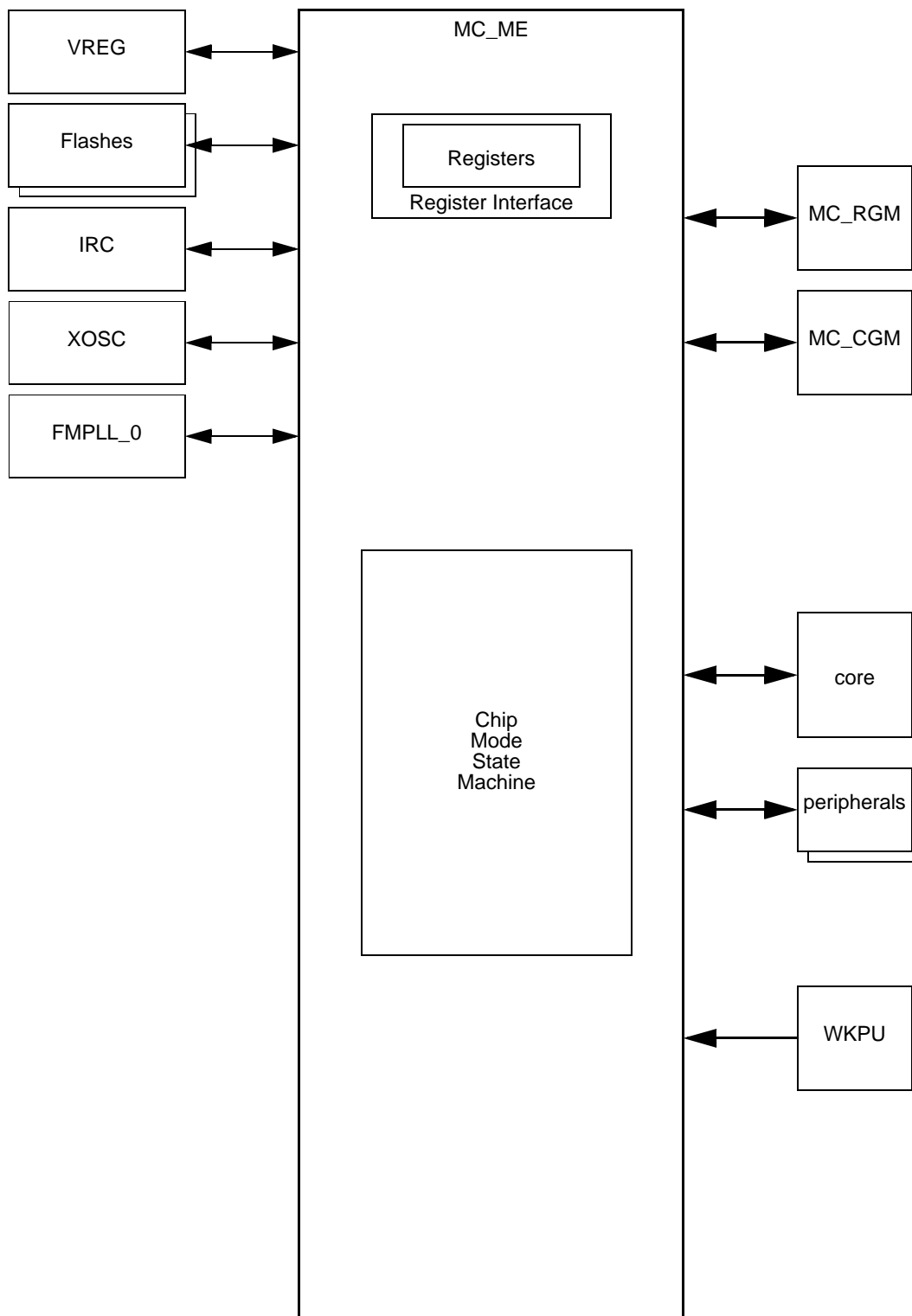


Figure 34. MC_ME Block Diagram

6.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the **ME_ME** register
- definition of various chip mode configurations by the **ME_<mode>_MC** registers
- control of the actual chip mode by the **ME_MCTL** register
- capture of the current mode and various resource status within the contents of the **ME_GS** register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the **ME_RUN_PC0...7**, **ME_LP_PC0...7**, and **ME_PCTLn** registers
- capture of current peripheral clock gated/enabled status

6.1.3 Modes of Operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as **RESET**, **DRUN**, **SAFE**, and **TEST**. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as **RUN0...3**, **HALT0**, and **STOP0** which can be configured to meet the application requirements in terms of energy management and available processing power. The modes **DRUN**, **SAFE**, **TEST**, and **RUN0...3** are the chip software running modes.

Table 35 describes the MC_ME modes.

Table 35. MC_ME Mode Descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAM when present is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE , TEST and RUN0...3	system reset assertion, RUN0...3 , TEST via software, SAFE via software or hardware failure.
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN , and RUN0...3	system reset assertion, DRUN via software

Table 35. MC_ME Mode Descriptions (continued)

Name	Description	Entry	Exit
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0...3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0...3 , interrupt event from HALT0 , interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0...3 modes, HALT0 , STOP0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency.	software request from RUN0...3	system reset assertion, SAFE on recoverable hardware failure, RUN0...3 on off-platform interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0...3	system reset assertion, SAFE on recoverable hardware failure, RUN0...3 on interrupt event or wakeup event

6.2 External Signal Description

The MC_ME has no connections to any external pins.

6.3 Memory Map and Register Definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting

6.3.1 Memory Map

Table 36. MC_ME Register Description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FD_C000	ME_GS	Global Status	word	read	read	read	on page 105
0xC3FD_C004	ME_MCTL	Mode Control	word	read	read/write	read/write	on page 107
0xC3FD_C008	ME_ME	Mode Enable	word	read	read/write	read/write	on page 108
0xC3FD_C00C	ME_IS	Interrupt Status	word	read	read/write	read/write	on page 110
0xC3FD_C010	ME_IM	Interrupt Mask	word	read	read/write	read/write	on page 111
0xC3FD_C014	ME_IMTS	Invalid Mode Transition Status	word	read	read/write	read/write	on page 112
0xC3FD_C018	ME_DMTS	Debug Mode Transition Status	word	read	read	read	on page 113
0xC3FD_C020	ME_RESET_MC	RESET Mode Configuration	word	read	read	read	on page 116
0xC3FD_C024	ME_TEST_MC	TEST Mode Configuration	word	read	read/write	read/write	on page 116
0xC3FD_C028	ME_SAFE_MC	SAFE Mode Configuration	word	read	read/write	read/write	on page 117
0xC3FD_C02C	ME_DRUN_MC	DRUN Mode Configuration	word	read	read/write	read/write	on page 117
0xC3FD_C030	ME_RUN0_MC	RUN0 Mode Configuration	word	read	read/write	read/write	on page 118
0xC3FD_C034	ME_RUN1_MC	RUN1 Mode Configuration	word	read	read/write	read/write	on page 118
0xC3FD_C038	ME_RUN2_MC	RUN2 Mode Configuration	word	read	read/write	read/write	on page 118
0xC3FD_C03C	ME_RUN3_MC	RUN3 Mode Configuration	word	read	read/write	read/write	on page 118
0xC3FD_C040	ME_HALT0_MC	HALT0 Mode Configuration	word	read	read/write	read/write	on page 118
0xC3FD_C048	ME_STOP0_MC	STOP0 Mode Configuration	word	read	read/write	read/write	on page 119
0xC3FD_C060	ME_PS0	Peripheral Status 0	word	read	read	read	on page 121
0xC3FD_C064	ME_PS1	Peripheral Status 1	word	read	read	read	on page 121

Table 36. MC_ME Register Description (continued)

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FD_C068	ME_PS2	Peripheral Status 2	word	read	read	read	on page 122
0xC3FD_C06C	ME_PS3	Peripheral Status 3	word	read	read	read	on page 122
0xC3FD_C080	ME_RUN_PC0	Run Peripheral Configuration 0	word	read	read/write	read/write	on page 123
0xC3FD_C084	ME_RUN_PC1	Run Peripheral Configuration 1	word	read	read/write	read/write	on page 123
...							
0xC3FD_C09C	ME_RUN_PC7	Run Peripheral Configuration 7	word	read	read/write	read/write	on page 123
0xC3FD_C0A0	ME_LP_PC0	Low-Power Peripheral Configuration 0	word	read	read/write	read/write	on page 124
0xC3FD_C0A4	ME_LP_PC1	Low-Power Peripheral Configuration 1	word	read	read/write	read/write	on page 124
...							
0xC3FD_C0BC	ME_LP_PC7	Low-Power Peripheral Configuration 7	word	read	read/write	read/write	on page 124
0xC3FD_C0C4	ME_PCTL4	DSPI0 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0C5	ME_PCTL5	DSPI1 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0C6	ME_PCTL6	DSPI2 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0D0	ME_PCTL16	FlexCAN0 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0D6	ME_PCTL22	SAI0 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0D7	ME_PCTL23	DMA_CH_MUX Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0DC	ME_PCTL28	SAI1 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0DD	ME_PCTL29	SAI2 Control	byte	read	read/write	read/write	on page 124
0xC3FD_CODE	ME_PCTL30	Video Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0E0	ME_PCTL32	ADC0 Control	byte	read	read/write	read/write	on page 124

Table 36. MC_ME Register Description (continued)

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FD_C0E6	ME_PCTL38	eTimer0 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0EC	ME_PCTL44	I2C_DMA0 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0ED	ME_PCTL45	I2C_DMA1 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0F0	ME_PCTL48	LIN_FLEX0 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0F1	ME_PCTL49	LIN_FLEX1 Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0FA	ME_PCTL58	CRC Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0FD	ME_PCTL61	PTP Control	byte	read	read/write	read/write	on page 124
0xC3FD_C0FE	ME_PCTL62	CE_RTC Control	byte	read	read/write	read/write	on page 124
0xC3FD_C11C	ME_PCTL92	PIT_RTI Control	byte	read	read/write	read/write	on page 124
0xC3FD_C128	ME_PCTL104	CMU0 Control	byte	read	read/write	read/write	on page 124

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 37. MC_ME Memory Map

Address	Name	0		1		2		3		27		5		6		7		8		9		10		11		12		13		14		15								
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																							
0xC3FD_C000	ME_GS	R	S_CURRENT_MODE				S_MTRANS	1	0	0	S_PDO	0	0	S_MVR	S_DFLA		S_CFLA																							
		W																																						
		R											S_FMPLL_0	S_XOSC	S_IRC	S_SYSCLK																								
		W																																						
0xC3FD_C004	ME_MCTL	R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
		W																																						
		R	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
		W	KEY																																					
0xC3FD_C008	ME_ME	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
		W																																						
		R					0	0	0	0	0	STOP0	0	0	HALT0	0	0	RUN3	0	0	RUN2	0	0	RUN1	0	0	RUN0	0	0	DRUN	0	0	SAFE	0	0	TEST	0	0	RESET_FUNC	
		W																																						
0xC3FD_C00C	ME_IS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
		W																																						
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	L_ICONF_CU	L_ICONF	L_IMODE	L_SAFE	L_MTC										
		W																							w1c	w1c	w1c	w1c	w1c											
0xC3FD_C010	ME_IM	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
		W																																						
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC										
		W																							M_ICONF	M_IMODE	M_SAFE	M_MTC												

Table 37. MC_ME Memory Map (continued)

Address	Name	Bit Positions																
		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0xC3FD_C014	ME_IMTS	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
		W												w1c	w1c	w1c	w1c	w1c
0xC3FD_C018	ME_DMTS	R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	CORE_DBG	0	0	SMR
		W																
		R	0	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSClk_SW	DFLASH_SC	CFLASH_SC	CDP_PRRH_0_143	0	0		CDP_PRRH_96_127	CDP_PRRH_64_95	CDP_PRRH_32_63	CDP_PRRH_0_31
		W																
0xC3FD_C01C	reserved																	
0xC3FD_C020	ME_RESET_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON	CFLAON		
		W																
		R										FMPLL_00N	XOSCON	IRCON	SYSClk			
		W																
0xC3FD_C024	ME_TEST_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON	CFLAON		
		W																
		R	0	0	0	0	0	0	0	0	0	FMPLL_00N	XOSCON	IRCON	SYSClk			
		W																

Table 37. MC_ME Memory Map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0xC3FD_C028	ME_SAFE_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON		
		W																	
		R											FMPLL_00N	XOSCON	IRCON	SYSCLK			
		W																	
0xC3FD_C02C	ME_DRUN_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON		
		W																	
		R	0	0	0	0	0	0	0	0	0		FMPLL_00N	XOSCON	IRCON	SYSCLK			
		W																	
0xC3FD_C030 ... 0xC3FD_C03C	ME_RUN0...3_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON		
		W																	
		R	0	0	0	0	0	0	0	0	0		FMPLL_00N	XOSCON	IRCON	SYSCLK			
		W																	
0xC3FD_C040	ME_HALTO_MC	R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON		
		W																	
		R	0	0	0	0	0	0	0	0	0		FMPLL_00N	XOSCON	IRCON	SYSCLK			
		W																	
0xC3FD_C044	reserved																		

Table 37. MC_ME Memory Map (continued)

Address	Name	0		1		2		3		27		5		6		7		8		9		10		11		12		13		14		15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
0xC3FD_C048	ME_STOP0_MC	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDO	0	0	0	0	MVRON	DFLAON		CFLAON						
		W																															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		OSCON			IRCON	SYSCLK								
		W																															
0xC3FD_C04C ... 0xC3FD_C05C	reserved																																
0xC3FD_C060	ME_PS0	R		S_Video	S_SAI2	S_SAI1													S_DMA_CH_MUX	S_SAI0												S_FlexCAN0	
		W																															
		R																			S_DSP12	S_DSP11	S_DSP10										
		W																															
0xC3FD_C064	ME_PS1	R		S_CE_RTC	S_PTP					S_CRC																			S_LIN_FLEX1	S_LIN_FLEX0			
		W																															
		R			S_I2C_DMA1	S_I2C_DMA0														S_eTimer0											S_ADC0		
		W																															

Table 37. MC_ME Memory Map (continued)

Address	Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
			16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0xC3FD_C068	ME_PS2	R				S_PIT_RTI													
		W																	
		R																	
		W																	
0xC3FD_C06C	ME_PS3	R																	
		W																	
		R									S_CMU0								
		W																	
0xC3FD_C070	reserved																		
0xC3FD_C074 ... 0xC3FD_C07C	reserved																		
0xC3FD_C080 ... 0xC3FD_C09C	ME_RUN_PC 0...7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	0	0	0		RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
		W																	
0xC3FD_C0A0 ... 0xC3FD_C0BC	ME_LP_PC0 ...7	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																	
		R	0	0	0	0	0	STOP0		0	HALT0	0	0	0	0	0	0	0	0
		W																	

Table 37. MC_ME Memory Map (continued)

Address	Name	0		1		2		3		27		5		6		7		8		9		10		11		12		13		14		15																																
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																															
0xC3FD_C0C0 ...	ME_PCTL0... 143 ¹	R	0	DBG_F	LP_CFG				RUN_CFG				0	DBG_F	LP_CFG				RUN_CFG																																													
		W		DBG_F										DBG_F																																																		
0xC3FD_C14C		R	0	DBG_F	LP_CFG				RUN_CFG				0	DBG_F	LP_CFG				RUN_CFG																																													
		W		DBG_F										DBG_F																																																		
0xC3FD_C150 ...	reserved																																																															
0xC3FD_FFFC																																																																

¹ There is space in the register map for 144 peripherals. Please refer to Table 36 for the ME_PCTLn locations actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

6.3.2 Register Description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0xC3FD_C080, as a half-word at address 0xC3FD_C082, or as a byte at address 0xC3FD_C083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

6.3.2.1 Global Status Register (ME_GS)

Address 0xC3FD_C000 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	1	0	0	S_PDO	0	0	S_MVR	S_DFLA		S_CFLA	
W																
Reset	0	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R										S_FMPLL_0	S_XOSC	S_IRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 35. Global Status Register (ME_GS)

This register contains global mode status.

Table 38. Global Status Register (ME_GS) Field Descriptions

Field	Description
S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
S_PDO	Output power-down status — This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and the pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
S_DFLA	Data flash availability status 00 Data flash is not available 01 Data flash is in power-down mode 10 Data flash is not available 11 Data flash is in normal mode and available for use
S_CFLA	Code flash availability status 00 Code flash is not available 01 Code flash is in power-down mode 10 Code flash is in low-power mode 11 Code flash is in normal mode and available for use
S_FMPLL_0	system PLL status 0 system PLL is not stable 1 system PLL is providing a stable clock
S_XOSC	external oscillator status 0 external oscillator is not stable 1 external oscillator is providing a stable clock

Table 38. Global Status Register (ME_GS) Field Descriptions (continued)

Field	Description
S_IRC	internal RC oscillator status 0 internal RC oscillator is not stable 1 internal RC oscillator is providing a stable clock
S_SYSCLK	System clock switch status — These bits specify the system clock currently used by the system. 0000 IRC 0001 reserved 0010 XOSC 0011 reserved 0100 FMPLL_0 PCS 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

6.3.2.2 Mode Control Register (ME_MCTL)

Address 0xC3FD_C004 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
W	KEY															
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 36. Mode Control Register (ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by **ME_ME** register bits, configurations corresponding to unavailable modes are reserved and access to **ME_<mode>_MC** registers must respect this for successful mode requests.

NOTE

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Table 39. Mode Control Register (ME_MCTL) Field Descriptions

Field	Description
TARGET_M ODE	<p>Target chip mode — These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0...3 mode value.</p> <p>0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 disabled 1110 reserved 1111 disabled</p> <p>Note: 1101 and 1111 modes are permanently disabled. Setting these modes will set S_DMA bit in ME_IMTS register.</p>
KEY	<p>Control key — These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key.</p> <p>KEY:0101101011110000 (0x5AF0) INVERTED KEY:1010010100001111 (0xA50F)</p>

6.3.2.3 Mode Enable Register (ME_ME)

Address 0xC3FD_C008 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R						STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	01

Figure 37. Mode Enable Register (ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. **RESET**, **SAFE**, **DRUN**, and **RUN0** modes are always enabled.

Table 40. Mode Enable Register (ME_ME) Field Descriptions

Field	Description
STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
RUN0	RUN0 mode enable 1 RUN0 mode is enabled
DRUN	DRUN mode enable 1 DRUN mode is enabled
SAFE	SAFE mode enable 1 SAFE mode is enabled
TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
RESET_FUN C	'functional' RESET mode enable 1 'functional' RESET mode is enabled

6.3.2.4 Interrupt Status Register (ME_IS)

Address 0xC3FD_C00C Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	I_ICONF_CU	I_ICONF	I_IMODE	I_SAFE	I_MTC
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38. Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Table 41. Interrupt Status Register (ME_IS) Field Descriptions

Field	Description
I_ICONF_CU	<p>Invalid mode configuration interrupt (Clock Usage) — This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a ‘1’ to this bit.</p> <p>0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending</p>
I_ICONF	<p>Invalid mode configuration interrupt — This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a ‘1’ to this bit.</p> <p>0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending</p>
I_IMODE	<p>Invalid mode interrupt — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a ‘1’ to this bit.</p> <p>0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending</p>
I_SAFE	<p>SAFE mode interrupt — This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a ‘1’ to this bit.</p> <p>0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending</p>
I_MTC	<p>Mode transition complete interrupt — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a ‘1’ to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0.</p> <p>0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending</p>

6.3.2.5 Interrupt Mask Register (ME_IM)

Address 0xC3FD_C010				Access: User read, Supervisor read/write, Test read/write													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 39. Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Table 42. Interrupt Mask Register (ME_IM) Field Descriptions

Field	Description
M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

6.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

Address 0xC3FD_C014 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40. Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Table 43. Invalid Mode Transition Status Register (ME_IMTS) Field Descriptions

Field	Description
S_MTI	Mode Transition Illegal status — This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is '1'). Please refer to Section 6.4.5, "Mode Transition Interrupts" for the exceptions to this behavior. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
S_MRI	Mode Request Illegal status — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
S_DMA	Disabled Mode Access status — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
S_NMA	Non-existing Mode Access status — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_MCTL register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
S_SEA	SAFE Event Active status — This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

6.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

Address 0xC3FD_C018 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	CORE_DBG	0	0	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	DFLASH_SC	CFLASH_SC	CDP_PRPH_0_143	0	0		CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41. Debug Mode Transition Status Register (ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by **ME_GS.S_MTRANS** may be taking longer than expected.

NOTE

The **ME_DMTS** register does not indicate whether a mode transition is ongoing. Therefore, some **ME_DMTS** bits may still be asserted after the mode transition has completed.

Table 44. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions

Field	Description
PREVIOUS_MODE	Previous chip mode — These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOPO 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator — This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet responded. It is cleared when the MC_PCU has responded. 0 Handshake is not busy 1 Handshake is busy
PMC_PROG	MC_PCU Mode Change in Progress indicator — This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
CORE_DBG	Processor is in Debug mode indicator — This bit is set while the processor is in debug mode. 0 The processor is not in debug mode 1 The processor is in debug mode
SMR	SAFE mode request from MC_RGM is active indicator — This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
VREG_CSR_C_SC	Main VREG dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
CSRC_CSR_C_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place

Table 44. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions (continued)

Field	Description
IRC_SC	IRC State Change during mode transition indicator — This bit is set when the internal RC oscillator is requested to change its power up/down state. It is cleared when the internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
SYSCLK_S W	System Clock Switching pending status — 0 No system clock source switching is pending 1 A system clock source switching is pending
DFLASH_SC	DFLASH State Change during mode transition indicator — This bit is set when the DFLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
CFLASH_SC	CFLASH State Change during mode transition indicator — This bit is set when the CFLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
CDP_PRPH _0_143	Clock Disable Process Pending status for Peripherals 0...143 ¹ — This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH _96_127	Clock Disable Process Pending status for Peripherals 96...127 ¹ — This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH _64_95	Clock Disable Process Pending status for Peripherals 64...95 ¹ — This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH _32_63	Clock Disable Process Pending status for Peripherals 32...63 ¹ — This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
CDP_PRPH _0_31	Clock Disable Process Pending status for Peripherals 0...31 ¹ — This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

¹ Peripheral *n* corresponds to the **ME_PCTL*n*** register. Please refer to [Table 36](#) for the **ME_PCTL*n*** locations actually occupied, which in turn indicates which peripherals are reported in the **ME_DMITS** register.

6.3.2.8 RESET Mode Configuration Register (ME_RESET_MC)

Address 0xC3FD_C020 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	[Greyed out]				[Greyed out]				[Greyed out]	FMPLL_OON	XOSCON	IRCON	SYSCLK			
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 42. RESET Mode Configuration Register (ME_RESET_MC)

This register configures system behavior during **RESET** mode. Please refer to [Table 45](#) for details.

6.3.2.9 TEST Mode Configuration Register (ME_TEST_MC)

Address 0xC3FD_C024 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FMPLL_OON	XOSCON	IRCON	SYSCLK			
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 43. TEST Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during **TEST** mode. Please refer to [Table 45](#) for details.

NOTE

Byte write accesses are not allowed to this register.

6.3.2.10 SAFE Mode Configuration Register (ME_SAFE_MC)

Address 0xC3FD_C028 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Shaded]									[Shaded]			[Shaded]			
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	[Shaded]				[Shaded]					FMPLL_OON	XOSCON	IRCON	SYSCLK			
W	[Shaded]				[Shaded]								[Shaded]			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 44. SAFE Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during **SAFE** mode. Please refer to [Table 45](#) for details.

NOTE

Byte write accesses are not allowed to this register.

6.3.2.11 DRUN Mode Configuration Register (ME_DRUN_MC)

Address 0xC3FD_C02C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Shaded]									[Shaded]			[Shaded]			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FMPLL_OON	XOSCON	IRCON	SYSCLK			
W	[Shaded]				[Shaded]								[Shaded]			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 45. DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during **DRUN** mode. Please refer to [Table 45](#) for details.

NOTE

Byte write accesses are not allowed to this register.

6.3.2.12 RUN0..3 Mode Configuration Register (ME_RUN0..3_MC)

Address 0xC3FD_C030 - 0xC3FD_C03C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FMPLL_OON	XOSCON	IRCON	SYSCLK			
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 46. RUN0...3 Mode Configuration Registers (ME_RUN0...3_MC)

This register configures system behavior during **RUN0...3** modes. Please refer to [Table 45](#) for details.

NOTE

Byte write accesses are not allowed to this register.

6.3.2.13 HALT0 Mode Configuration Register (ME_HALT0_MC)

Address 0xC3FD_C040 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FMPLL_OON	XOSCON	IRCON	SYSCLK			
W	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 47. HALT0 Mode Configuration Register (ME_HALT0_MC)

This register configures system behavior during **HALT0** mode. Please refer to [Table 45](#) for details.

NOTE

Byte write accesses are not allowed to this register.

6.3.2.14 STOP0 Mode Configuration Register (ME_STOP0_MC)

Address 0xC3FD_C048 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	PDO	0	0	MVRON	DFLAON		CFLAON	
W	[Shaded]									[Shaded]			[Shaded]		[Shaded]	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	FMPLL_00N ¹	XOSCON	IRCON	SYSCLK			
W	[Shaded]									[Shaded]		[Shaded]	[Shaded]			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 48. STOP0 Mode Configuration Register (ME_STOP0_MC)

¹ Invalid mode configuration interrupt (L_ICONF) is generated if software tries to set this bit.

This register configures system behavior during **STOP0** mode. Please refer to [Table 45](#) for details.

NOTE

Byte write accesses are not allowed to this register.

Table 45. Mode Configuration Registers (ME_<mode>_MC) Field Descriptions

Field	Description
PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
DFLAON	Data flash power-down control — This bit specifies the operating mode of the data flash after entering this mode. 00 reserved 01 reserved 10 reserved 11 Data flash is in normal mode Note: Data flash should be kept in Normal mode in Stop and Halt mode.

Table 45. Mode Configuration Registers (ME_<mode>_MC) Field Descriptions (continued)

Field	Description
CFLAON	Code flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 reserved 10 reserved 11 Code flash is in normal mode Note: Code flash should be kept in Normal mode in Stop and Halt mode.
FMPLL_0ON	System PLL control 0 system PLL is switched off 1 system PLL is switched on
XOSCON	external oscillator control 0 external oscillator is switched off 1 external oscillator is switched on
IRCON	internal RC oscillator control 0 internal RC oscillator is switched off 1 internal RC oscillator is switched on
SYSClk	System clock switch control — These bits specify the system clock to be used by the system. 0000 IRC 0001 reserved 0010 XOSC 0011 reserved 0100 FMPLL_0 PCS 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

6.3.2.15 Peripheral Status Register 0 (ME_PS0)

Address 0xC3FD_C060 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		S_Video	S_SAI2	S_SAI1					S_DMA_CH_MUX	S_SAI0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R										S_DSP12	S_DSP11	S_DSP10				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 49. Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. Please refer to [Table 46](#) for details.

6.3.2.16 Peripheral Status Register 1 (ME_PS1)

Address 0xC3FD_C064 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		S_CE_RTC	S_PTP			S_CRC									S_LIN_FLEX1	S_LIN_FLEX0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			S_I2C_DMA1	S_I2C_DMA0						S_eTimer0						S_ADC0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 50. Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. Please refer to [Table 46](#) for details.

6.3.2.17 Peripheral Status Register 2 (ME_PS2)

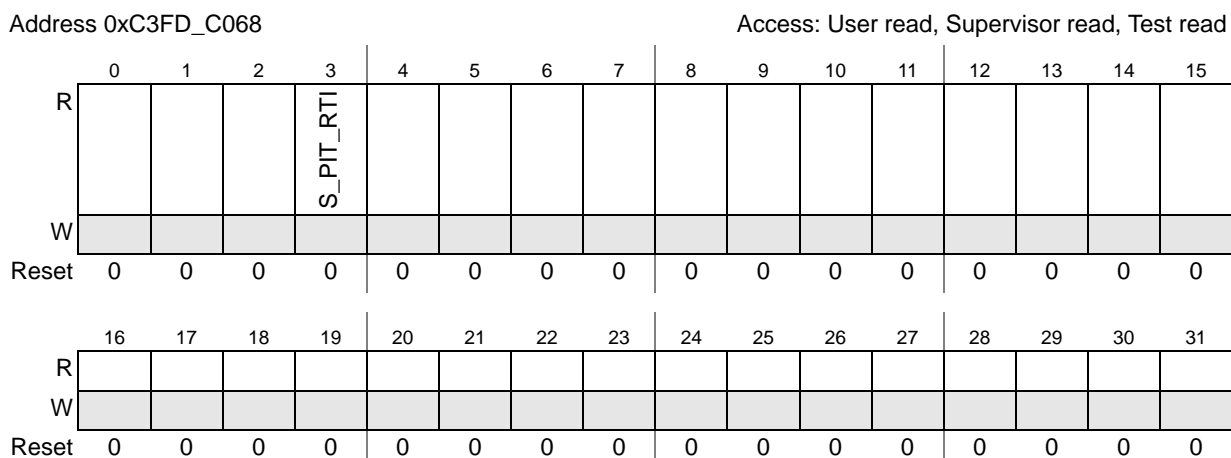


Figure 51. Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. Please refer to [Table 46](#) for details.

6.3.2.18 Peripheral Status Register 3 (ME_PS3)

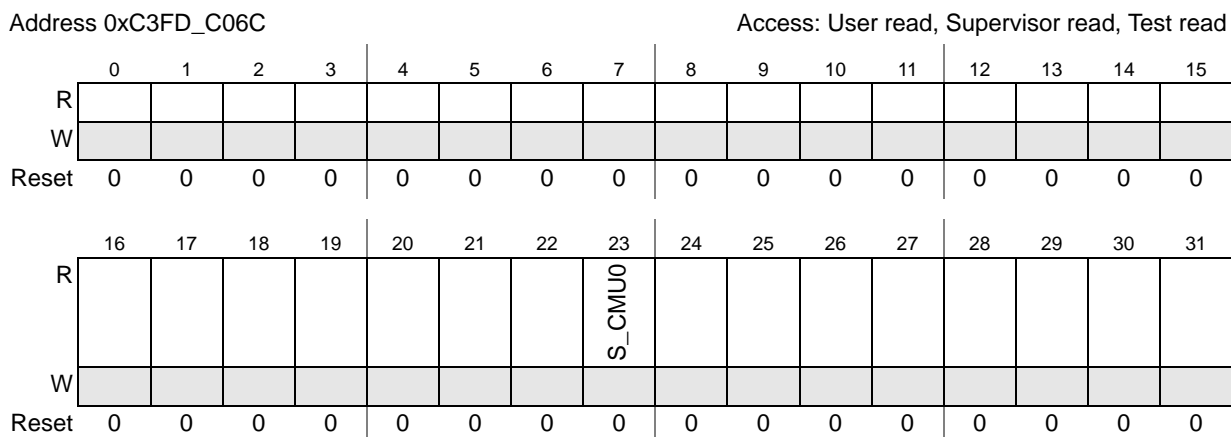


Figure 52. Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. Please refer to [Table 46](#) for details.

Table 46. Peripheral Status Registers (ME_PS n) Field Descriptions

Field	Description
S_<periph>	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

6.3.2.19 Run Peripheral Configuration Registers (ME_RUN_PC0...7)

Address 0xC3FD_C080 - 0xC3FD_C09C								Access: User read, Supervisor read/write, Test read/write								
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W									RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 53. Run Peripheral Configuration Registers (ME_RUN_PC0...7)

These registers configure eight different types of peripheral behavior during run modes.

Table 47. Run Peripheral Configuration Registers (ME_RUN_PC0...7) Field Descriptions

Field	Description
RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated 1 Peripheral is active
RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

6.3.2.20 Low-Power Peripheral Configuration Registers (ME_LP_PC0...7)

Address 0xC3FD_C0A0 - 0xC3FD_C0BC Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	STOP0	0	HALT0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 54. Low-Power Peripheral Configuration Registers (ME_LP_PC0...7)

These registers configure eight different types of peripheral behavior during non-run modes.

Table 48. Low-Power Peripheral Configuration Registers (ME_LP_PC0...7) Field Descriptions

Field	Description
STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active

6.3.2.21 Peripheral Control Registers (ME_PCTLn)

Address 0xC3FD_C0C0 - 0xC3FD_C14F Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0							
W		DBG_F		LP_CFG			RUN_CFG	
Reset	0	0	0	0	0	0	0	0

Figure 55. Peripheral Control Registers (ME_PCTLn)

These registers select the configurations during run and non-run modes for each peripheral. Please refer to [Table 36](#) for information on which **ME_PCTLn** locations are actually occupied. The unoccupied locations contain a read-only byte value of 0x00.

Table 49. Peripheral Control Registers (ME_PCTLn) Field Descriptions

Field	Description
DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes. <p style="text-align: center;">NOTE</p> This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.
LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0...7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0...7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

NOTE

After modifying any of the **ME_RUN_PC0...7**, **ME_LP_PC0...7**, and **ME_PCTLn** registers, software must request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behavior between the peripheral clock control process and the clock status reporting in the **ME_PSn** registers.

6.4 Functional Description

6.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register **ME_MCTL**. But in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the **ME_MCTL** register twice by writing

Mode Entry Module (MC_ME)

- the first time with the value of the key (0x5AF0) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field,
- and the second time with the inverted value of the key (0xA50F) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding **ME_<mode>_MC** register. The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the **S_CURRENT_MODE** bit field and the **S_MTRANS** bit of the global status register **ME_GS** to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to [Section 6.4.5, “Mode Transition Interrupts”](#).

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as **RUN0...3** → **RUN0...3**, **DRUN** → **DRUN**, **SAFE** → **SAFE**, and **TEST** → **TEST** are considered valid mode transition requests. As soon as the mode request is accepted as valid, the **S_MTRANS** bit is set till the status in the **ME_GS** register matches the configuration programmed in the respective **ME_<mode>_MC** register.

NOTE

It is recommended that software poll the **S_MTRANS** bit in the **ME_GS** register after requesting a transition to **HALT0** or **STOP0** modes.

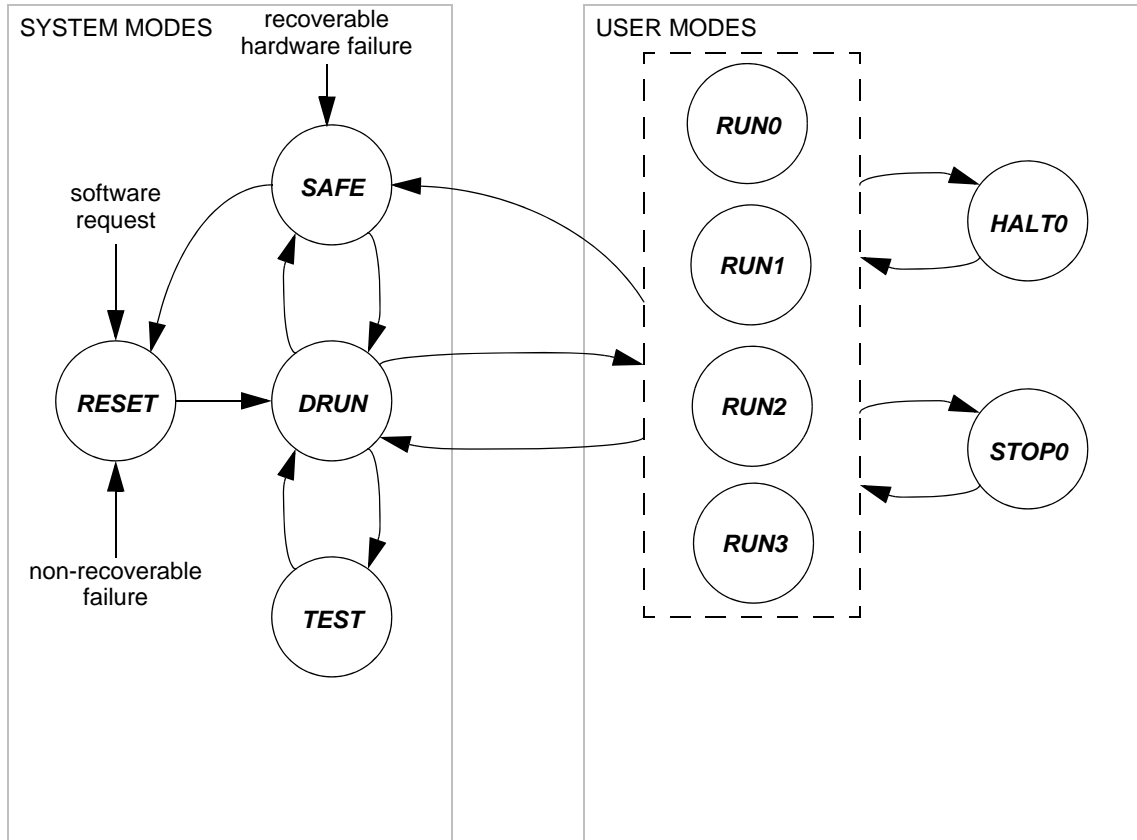


Figure 56. MC_ME Mode Diagram

6.4.2 Modes Details

6.4.2.1 RESET MODE

The chip enters this mode on the following events:

- from **SAFE**, **DRUN**, **RUN0...3**, or **TEST** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0000” for a ‘functional’ reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the **ME_RESET_MC** register. This mode has a pre-defined configuration, and the IRC is selected as the system clock.

6.4.2.2 DRUN Mode

The chip enters this mode on the following events:

- automatically from **RESET** mode after completion of the reset sequence

- from **RUN0...3**, **SAFE**, or **TEST** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0011”

As soon as any of the above events has occurred, a **DRUN** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_DRUN_MC** register. In this mode, the flashes, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the IRC as the system clock.

This mode is intended to be used by software

- to initialize all registers as per the system needs

NOTE

Software must ensure that the code executes from RAM before changing to this mode if the flashes are configured to be in the low-power or power-down state in this mode.

6.4.2.3 SAFE Mode

The chip enters this mode on the following events:

- from **DRUN**, **RUN0...3** when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0010”
- from any mode except **RESET** due to a **SAFE** mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

NOTE

If a hardware **SAFE** mode request occurs during **RESET**, depending on the timing of the **SAFE** mode request, **SAFE** mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after **DRUN** entry. The **SAFE** mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a **SAFE** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_SAFE_MC** register. This mode has a pre-defined configuration, and the IRC is selected as the system clock.

If the **SAFE** mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the **SAFE** mode regardless of other pending requests or new requests during the mode transition. Any new mode request made during a transition to the **SAFE** mode will cause an invalid mode interrupt.

NOTE

If software requests to change to the **SAFE** mode and then requests to change back to the parent mode before the mode transition is completed, the chip’s final mode after mode transition will be the **SAFE** mode.

As long as a **SAFE** event is active, the system remains in the **SAFE** mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
 - re-initialize the chip via the **DRUN** mode, or
 - completely reset the chip via the **RESET** mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the **PDO** bit of the **ME_SAFE_MC** register should be set. The input levels remain unchanged.

6.4.2.4 Test Mode

The chip enters this mode on the following event:

- from the **DRUN** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0001”

As soon as the above event has occurred, a **TEST** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_TEST_MC** register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the **SYSCLK** bit field to “1111”, and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

NOTE

Software must ensure that the code executes from RAM before changing to this mode if the flashes are configured to be in the low-power or power-down state in this mode.

6.4.2.5 RUN0..3 Modes

The chip enters one of these modes on the following events:

- from the **DRUN**, **SAFE**, or another **RUN0...3** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0100...0111”
- from the **HALT0** mode due to an off-platform interrupt event
- from the **STOP0** mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a **RUN0...3** mode transition request is generated. The mode configuration information for these modes is provided by the **ME_RUN0...3_MC** registers. In these modes, the flashes, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

NOTE

Software must ensure that the code executes from RAM before changing to this mode if the flashes are configured to be in the low-power or power-down state in this mode.

6.4.2.6 HALT0 Mode

The chip enters this mode on the following event:

- from one of the **RUN0...3** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1000”.

As soon as the above event has occurred, a **HALT0** mode transition request is generated. The mode configuration information for this mode is provided by **ME_HALT0_MC** register. This mode is quite configurable, and the **ME_HALT0_MC** register should be programmed according to the system needs. The flashes can be put in low-power or power-down mode as needed. If there is a **HALT0** mode request while an interrupt request is active, the transition to **HALT0** is aborted with the resultant mode being the current mode, **SAFE** (on **SAFE** mode request), or **DRUN** (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clock frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

NOTE

It is good practice for software to ensure that the **S_MTRANS** bit in the **ME_GS** register has been cleared on **HALT0** mode exit to ensure that the previous **RUN0...3** mode configuration has been fully restored before executing critical code.

6.4.2.7 STOP0 Mode

The chip enters this mode on the following event:

- from one of the **RUN0...3** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1010”.

As soon as the above event has occurred, a **STOP0** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_STOP0_MC** register. This mode is fully configurable, and the **ME_STOP0_MC** register should be programmed according to the system needs.

The flashes can be put in power-down mode as needed. If there is a **STOP0** mode request while any interrupt or wakeup event is active, the transition to **STOP0** is aborted with the resultant mode being the

current mode, **SAFE** (on **SAFE** mode request), or **DRUN** (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the **STOP0** mode, the internal RC oscillator clock is selected as the system clock until the target clock is available.

NOTE

It is good practice for software to ensure that the **S_MTRANS** bit in the **ME_GS** register has been cleared on **STOP0** mode exit to ensure that the previous **RUN0...3** mode configuration has been fully restored before executing critical code.

6.4.3 Mode Transition Process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

6.4.3.1 Target Mode Request

The target mode is requested by accessing the **ME_MCTL** register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the **TARGET_MODE** bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to [Section 6.4.5, “Mode Transition Interrupts”](#) for details.

In the case of mode transitions occurring because of hardware events such as a reset, a **SAFE** mode request, or interrupt requests and wakeup events to exit from low-power modes, the **TARGET_MODE** bit field of the **ME_MCTL** register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit **S_MTRANS** of the **ME_GS** register.

A **RESET** mode requested via the **ME_MCTL** register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The **RESET** mode request has the highest priority, and the MC_ME is kept in the **RESET** mode during the entire reset sequence.

The **SAFE** mode request has the next highest priority after reset. It can be generated either by software via the **ME_MCTL** register from all software running modes including **DRUN**, **RUN0...3**, and **TEST** or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

6.4.3.2 Target Mode Configuration Loading

On completion of the [Target Mode Request](#) step, the target mode configuration from the **ME_<target mode>_MC** register is loaded to start the resources (voltage sources, clock sources, flashes, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in . A ‘√’ indicates that a given resource is configurable for a given mode.

Table 50. MC_ME Resource Control Overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0...3	HALT0	STOP0
IRC	on	√ on	on	on	on	√ on	√ on
XOSC	off	√ off	off	√ off	√ on	√ off	√ off
FMPLL_0	off	√ off	off	√ off	√ on	√ off	off
CFLASH	normal	√ normal	normal	√ normal	√ normal	√ normal	√ normal
DFLASH	normal	√ normal	normal	√ normal	√ normal	√ normal	√ normal
MVREG	on	on	on	on	on	√ on	√ on

6.4.3.3 Peripheral Clocks Disable

On completion of the [Target Mode Request](#) step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers **ME_RUN_PC0...7** and **ME_LP_PC0...7**, and the peripheral control registers **ME_PCTLn_**

NOTE

The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a **SAFE** mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The **SAFE** mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to [Section 6.4.6, “Peripheral Clock Gating”](#) for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the **SAFE** mode.

6.4.3.4 Processor Low-Power Mode Entry

If, on completion of the [Peripheral Clocks Disable](#) step, the mode transition is to the **HALT0** mode, the MC_ME requests the processor to enter its halted state. The processor acknowledges its halt state request after completing all outstanding bus transactions.

If, on completion of the [Peripheral Clocks Disable](#) step, the mode transition is to the **STOP0** mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request after completing all outstanding bus transactions.

6.4.3.5 Processor and System Memory Clock Disable

If, on completion of the [Processor Low-Power Mode Entry](#) step, the mode transition is to the **HALT0** or **STOP0** mode and the processor is in its appropriate halted or stopped state, the MC_ME disables the processor and system memory clocks to achieve further power saving.

The clocks to the processor and system memory are unaffected while transitioning between software running modes such as **DRUN**, **RUN0...3**, and **SAFE**.

WARNING

Clocks to the whole chip including the processor and system memory can be disabled in **TEST** mode.

6.4.3.6 Clock Sources Switch-On

On completion of the [Processor Low-Power Mode Entry](#) step, the MC_ME switches on all clock sources based on the **<clock source>ON** bits of the **ME_<current mode>_MC** and **ME_<target mode>_MC** registers. The following clock sources are switched on at this step:

- the internal RC oscillator

- the external oscillator
- the system PLL

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the **S_<clock source>** bits of **ME_GS** register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

6.4.3.7 Flash Modules Switch-On

On completion of the step, if one or more of the flashes needs to be switched to normal mode from its low-power or power-down mode based on the **CFLAON** and **DFLAON** bit fields of the **ME_<current mode>_MC** and **ME_<target mode>_MC** registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flashes are available for access, the **S_CFLA** and **S_DFLA** bit fields of the **ME_GS** register are updated to “11” by hardware.

WARNING

It is illegal to switch the CFLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

6.4.3.8 Pad Outputs-On

On completion of the step, if the **PDO** bit of the **ME_<target mode>_MC** register is cleared, then

- all pad outputs are enabled to return to their previous state
- the I/O pads power sequence driver is switched on

6.4.3.9 Peripheral Clocks Enable

Based on the current and target chip modes, the peripheral configuration registers **ME_RUN_PC0...7**, **ME_LP_PC0...7**, and the peripheral control registers **ME_PCTLn**, the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed.

6.4.3.10 Processor and Memory Clock Enable

If the mode transition is from any of the low-power modes **HALT0** or **STOP0** to **RUN0...3**, the clocks to the processor and system memory are enabled. The process of enabling these clocks is executed only after the [Flash Modules Switch-On](#) process is completed.

6.4.3.11 Processor Low-Power Mode Exit

If the mode transition is from any of the low-power modes **HALT0** or **STOP0** to **RUN0...3**, the MC_ME requests the processor to exit from its halted or stopped state. This step is executed only after the [Processor and Memory Clock Enable](#) process is completed.

6.4.3.12 System Clock Switching

Based on the **SYSCLK** bit field of the **ME_<current mode>_MC** and **ME_<target mode>_MC** registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the IRC takes effect only after the **S_IRC** bit of the **ME_GS** register is set by hardware (i.e., the internal RC oscillator has stabilized).
- The target clock configuration for the XOSC takes effect only after the **S_XOSC** bit of the **ME_GS** register is set by hardware (i.e., the external oscillator has stabilized).
- The target clock configuration for the FMPLL_0 PCS takes effect only after the **S_FMPLL_0** bit of the **ME_GS** register is set by hardware (i.e., the system PLL has stabilized).
- If the clock is to be disabled, the **SYSCLK** bit field should be programmed with “1111”. This is possible only in the **TEST** mode.

The current system clock configuration can be observed by reading the **S_SYSCLK** bit field of the **ME_GS** register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the [Clock Sources Switch-On](#) process has completed if the target system clock source is one of the following:
 - the internal RC oscillator
 - the system PLL
- the [Peripheral Clocks Disable](#) process has completed in order not to change the system clock frequency before peripherals close their internal activities

An overview of system clock source selection possibilities for each mode is shown in [Table 51](#). A ‘√’ indicates that a given clock source is selectable for a given mode.

Table 51. MC_ME System Clock Selection Overview

System Clock Source	Mode						
	<i>RESET</i>	<i>TEST</i>	<i>SAFE</i>	<i>DRUN</i>	<i>RUN0...3</i>	<i>HALT0</i>	<i>STOPO</i>
IRC	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)
XOSC		√		√	√	√	√
FMPLL_0 PCS		√		√	√	√	
system clock is disabled		√ ¹					

¹ disabling the system clock during **TEST** mode will require a reset in order to exit **TEST** mode

6.4.3.13 Pad Switch-Off

If the **PDO** bit of the **ME_<target mode>_MC** register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is **SAFE** or **TEST**

This step is executed only after the [Peripheral Clocks Disable](#) process has completed.

6.4.3.14 Clock Sources (with no Dependencies) Switch-Off

Based on the chip mode and the **<clock source>ON** bits of the **ME_<mode>_MC** registers, if a given clock source is to be switched off and no other clock source needs it to be on, the MC_ME requests the clock source to power down and updates its availability status bit **S_<clock source>** of the **ME_GS** register to '0'. The following clock sources switched off at this step:

- the system PLL

This step is executed only after the [System Clock Switching](#) process has completed.

6.4.3.15 Clock Sources (with Dependencies) Switch-Off

Based on the chip mode and the **<clock source>ON** bits of the **ME_<mode>_MC** registers, if a given clock source is to be switched off and all clock sources which need this clock source to be on have been switched off, the MC_ME requests the clock source to power down and updates its availability status bit **S_<clock source>** of the **ME_GS** register to '0'. The following clock sources switched off at this step:

- the external oscillator

This step is executed only after

- the [System Clock Switching](#) process has completed in order not to lose the current system clock during mode transition
- the [Clock Sources \(with no Dependencies\) Switch-Off](#) process has completed in order to, for example, prevent unwanted lock transitions

6.4.3.16 Flash Switch-Off

Based on the **CFLAON** and **DFLAON** bit fields of the **ME_<current mode>_MC** and **ME_<target mode>_MC** registers, if any of the flashes is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flashes is updated in the **S_CFLA** and **S_DFLA** bit fields of the **ME_GS** register. This step is executed only when the [Processor and System Memory Clock Disable](#) process has completed.

6.4.3.17 Current Mode Update

The current mode status bit field **S_CURRENT_MODE** of the **ME_GS** register is updated with the target mode bit field **TARGET_MODE** of the **ME_MCTL** register when :

- all the updated status bits in the **ME_GS** register match the configuration specified in the **ME_<target mode>_MC** register

- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (halt/stop) entry and exit processes are finished

NOTE

SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the **ME_GS.S_RC** bit to be set. This is to ensure that the **SAFE** mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the **S_MTRANS** bit of the **ME_GS** register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the **ME_DMTS** register can indicate which process is still in progress.

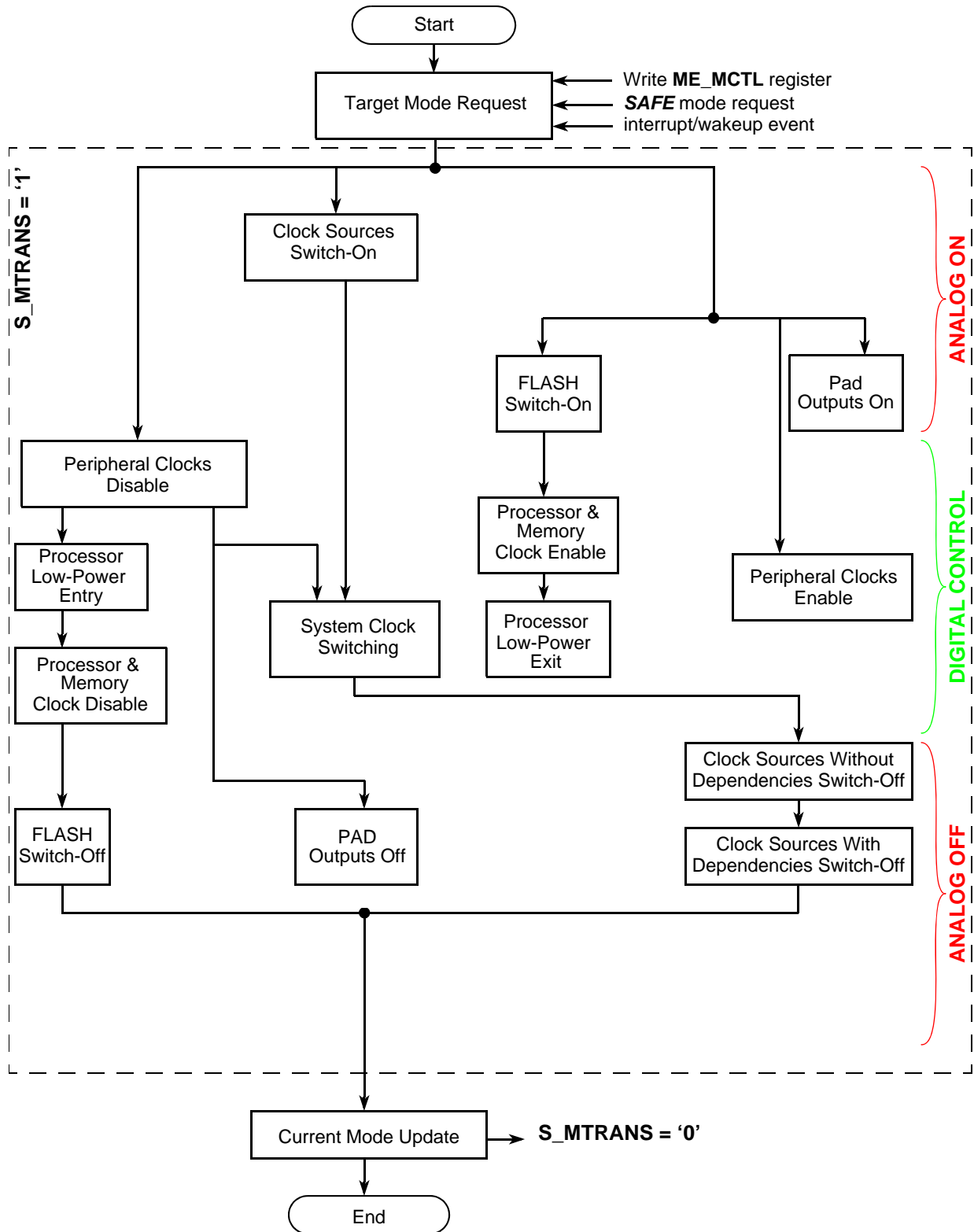


Figure 57. MC_ME Transition Diagram

6.4.4 Protection of Mode Configuration Registers

While programming the mode configuration registers **ME_<mode>_MC**, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the IRC is selected as the system clock, IRC must be on.
- If the XOSC clock is selected as the system clock, OSC must be on.
- If the FMPLL_0 PCS clock is selected as the system clock, PLL must be on.
- If FMPLL_0 is on, XOSC must also be on.

NOTE

Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration “00” for the **CFLAON** bit field is reserved.
- Configuration “00” for the **DFLAON** bit field is reserved.
- Configuration “10” for the **DFLAON** bit field is reserved.
- Configuration “11” for the **DFLAON** bit field with “01” or “10” for the **CFLAON** bit field is reserved.
- System clock configurations marked as ‘reserved’ may not be selected.
- Configuration “1111” for the **SYSCLK** bit field is allowed only for the **TEST** mode, and only in this case may all system clock sources be turned off.

WARNING

If the system clock is stopped during **TEST** mode, the chip can exit only via a system reset.

6.4.5 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a **SAFE** mode transition not due to a software request, and indicating when a mode transition has completed.

6.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the **ME_<mode>_MC** registers violating the protection rules mentioned in the [Section 6.4.4, “Protection of Mode Configuration Registers”](#), the interrupt pending bit **I_ICONF** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF** of the **ME_IM** register is ‘1’.

In addition, during a mode transition, if a clock source has been configured in the **ME_<target mode>_MC** register to be off and a peripheral requiring this clock source to be on has been enabled via the **ME_RUN_PC0...7/ME_LP_PC0...7** and **ME_PCTLn** registers, the interrupt pending

bit **I_ICONF_CU** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF_CU** of the **ME_IM** register is '1'.

6.4.5.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the **SAFE** mode and the **SAFE** mode request from MC_RGM is active, and if the target mode requested is other than **RESET** or **SAFE**, then this new mode request is considered to be invalid, and the **S_SEA** bit of the **ME_IMTS** register is set.
- If the **TARGET_MODE** bit field of the **ME_MCTL** register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the **S_NMA** bit of the **ME_IMTS** register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the **ME_MCTL** register.
- If some of the chip modes are disabled as programmed in the **ME_ME** register, their respective configurations are considered reserved, and any access to the **ME_MCTL** register with those values results in an invalid mode transition request. When such a disabled mode is requested, the **S_DMA** bit of the **ME_IMTS** register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the **ME_MCTL** register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit **S_MRI** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition is in progress (the **S_MTRANS** bit of the **ME_GS** register is '1'), the mode transition illegal status bit **S_MTI** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.

NOTE

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the **ME_IMTS** register in the order from highest to lowest is **S_SEA**, **S_NMA**, **S_DMA**, **S_MRI**, and **S_MTI**.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the **RESET** or **SAFE** mode irrespective of the mode transition status.
- As the exit of **HALT0** and **STOP0** modes depends on the interrupts of the system which can occur at any instant, these requests to return to **RUN0...3** modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

- Self-transition requests (e.g., **RUN0** → **RUN0**) are not considered as invalid even when the mode transition process is active (i.e., **S_MTRANS** is '1'). During the low-power mode exit process, if the system is not able to enter the respective **RUN0...3** mode properly (i.e., all status bits of the **ME_GS** register match with configuration bits in the **ME_<mode>_MC** register), then software can only request the **SAFE** or **RESET** mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit **I_IMODE** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_IMODE** of the **ME_IM** register is '1'.

6.4.5.3 SAFE Mode Transition Interrupt

Whenever the system enters the **SAFE** mode as a result of a **SAFE** mode request from the MC_RGM due to a hardware failure, the interrupt pending bit **I_SAFE** of the **ME_IS** register is set, and an interrupt is generated if the mask bit **M_SAFE** of **ME_IM** register is '1'.

The **SAFE** mode interrupt pending bit can be cleared only when the **SAFE** mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a **SAFE** mode request). If the system is already in **SAFE** mode, any new **SAFE** mode request by the MC_RGM also sets the interrupt pending bit **I_SAFE**. However, the **SAFE** mode interrupt pending bit is not set when the **SAFE** mode is entered by a software request (i.e., programming of **ME_MCTL** register).

6.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the **S_MTRANS** bit of **ME_GS** register transits from '1' to '0'), the interrupt pending bit **I_MTC** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_MTC** of the **ME_IM** register is '1'. The interrupt bit **I_MTC** is not set when entering low-power modes **HALT0** and **STOP0** in order to avoid the same event requesting the immediate exit of these low-power modes.

6.4.6 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers **ME_RUN_PC0...7** are chosen only during the software running modes **DRUN**, **TEST**, **SAFE**, and **RUN0...3**. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the **RUN_CFG** bit field of the **ME_PCTLn** registers.

The low-power peripheral configuration registers **ME_LP_PC0...7** are chosen only during the low-power modes **HALT0** and **STOP0**. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the **LP_CFG** bit field of the **ME_PCTLn** registers.

Any modifications to the **ME_RUN_PC0...7**, **ME_LP_PC0...7**, and **ME_PCTL n** registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the processor enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the **DBG_F** bit of the associated **ME_PCTL n** register is set. Otherwise, the peripheral clock gating status depends on the **RUN_CFG** and **LP_CFG** bits. Any further modifications of the **ME_RUN_PC0...7**, **ME_LP_PC0...7**, and **ME_PCTL n** registers during a debug session will take affect immediately without requiring any new mode request.

6.4.7 Application Example

[Figure 58](#) shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

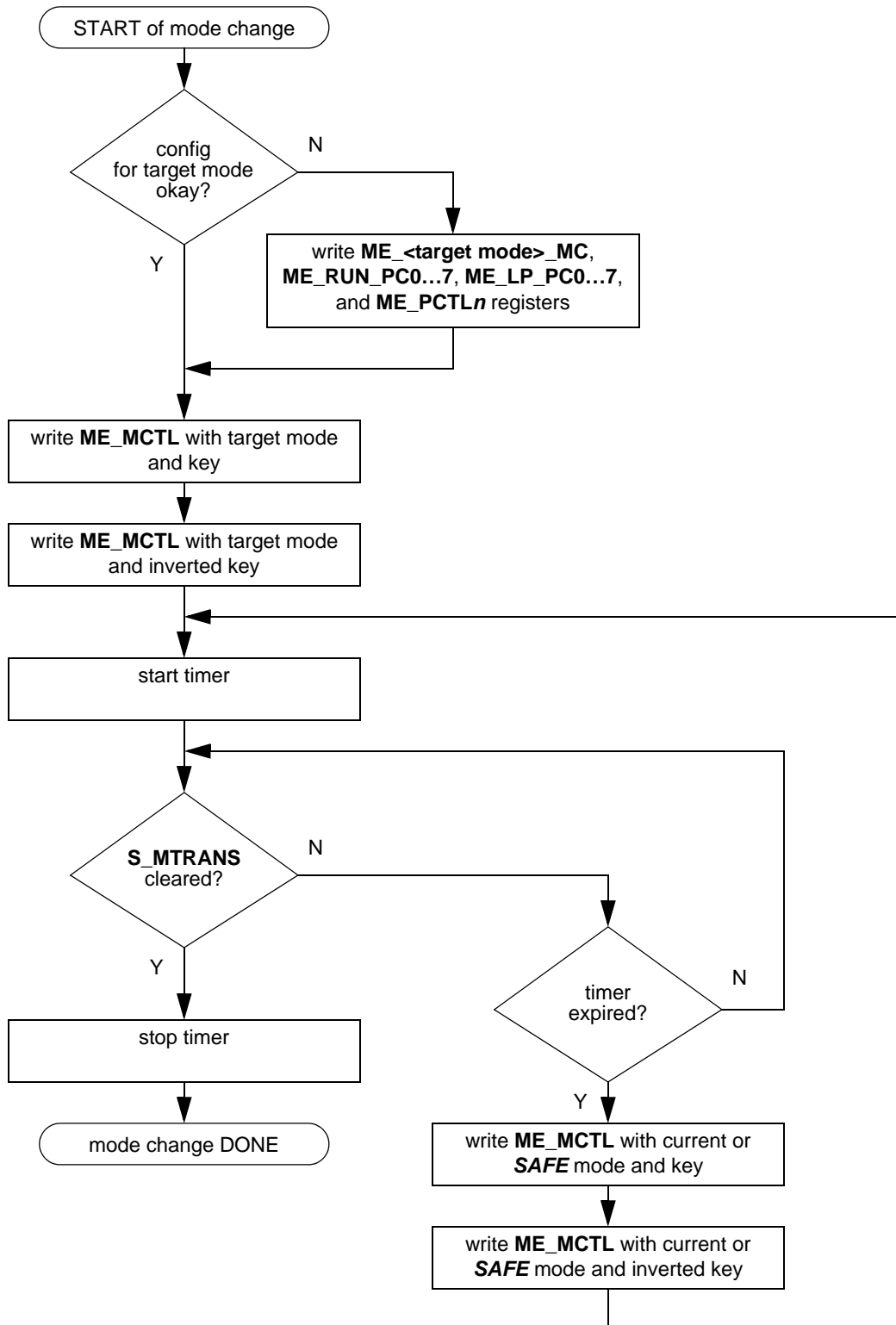


Figure 58. MC_ME Application Example Flow Diagram

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 7

Reset Generation Module (MC_RGM)

7.1 Introduction

7.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (***PHASE0***, ***PHASE1***, ***PHASE2***, ***PHASE3***, and ***IDLE***) of the reset sequence and controls the reset signals generated in the system.

[Figure 59](#) shows the MC_RGM block diagram.

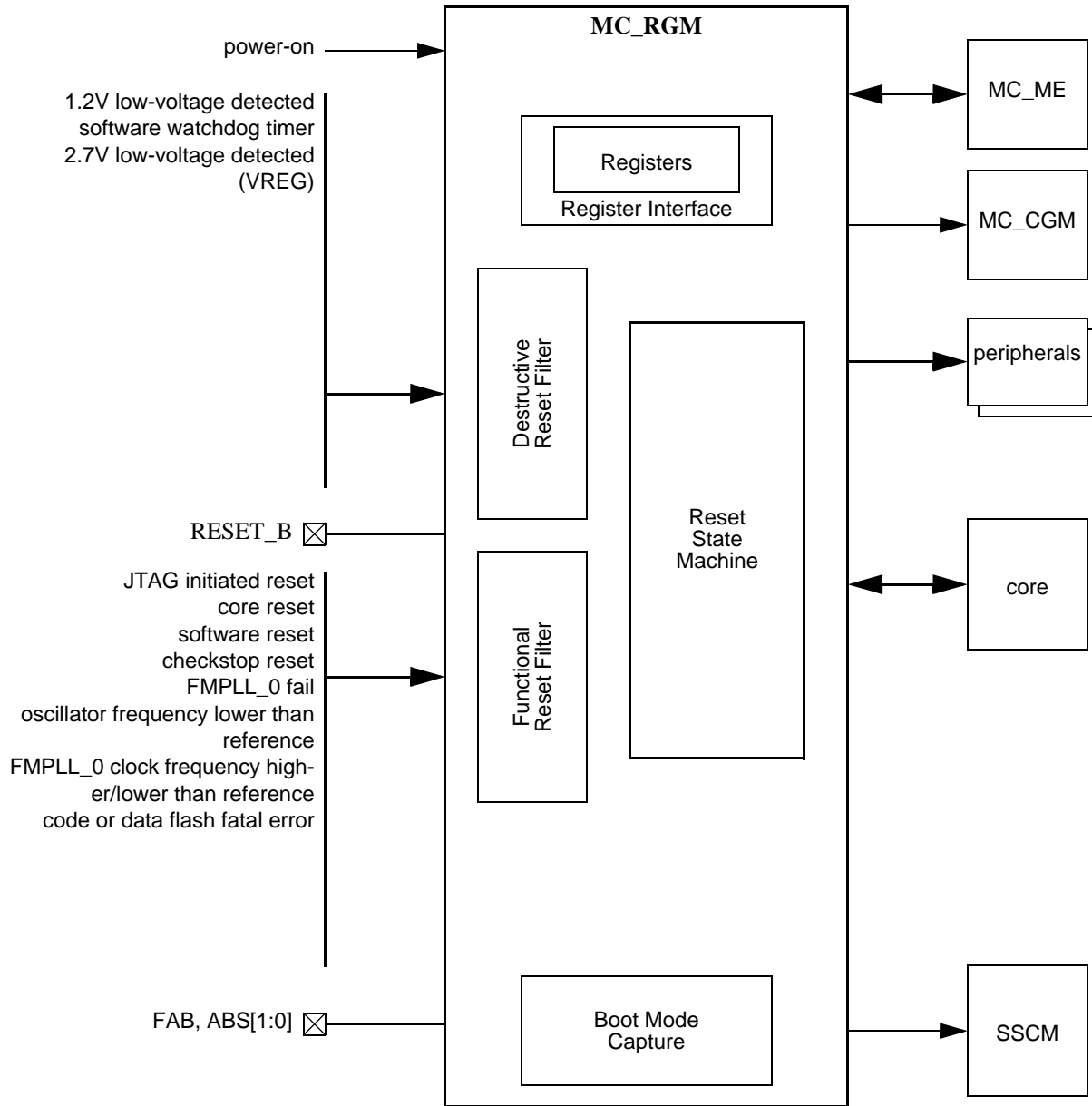


Figure 59. MC_RGM Block Diagram

7.1.2 Features

The MC_RGM contains the functionality for the following features:

- ‘destructive’ resets management
- ‘functional’ resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to **SAFE** mode or interrupt request events
- short reset sequence configuration

- bidirectional reset behavior configuration
- boot mode capture on RESET_B deassertion

7.1.3 Reset Sources

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

- A ‘destructive’ reset source is associated with an event related to a critical - usually hardware - error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the chip starting from **PHASE0**. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown. ‘Destructive’ resets are
 - power-on reset
 - 1.2V low-voltage detected
 - software watchdog timer
 - 2.7V low-voltage detected (VREG)
- A ‘functional’ reset source is associated with an event related to a less-critical - usually non-hardware - error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the chip starting from **PHASE1**. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (e.g., debug modules, flash modules) is preserved. ‘Functional’ resets are
 - external reset
 - JTAG initiated reset
 - core reset
 - software reset
 - checkstop reset
 - FMPLL_0 fail
 - oscillator frequency lower than reference
 - FMPLL_0 clock frequency higher/lower than reference
 - code or data flash fatal error

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., **PHASEn** states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the **IDLE** phase. During this entire process, the MC_ME state machine is held in **RESET** mode. Only at the end of the reset sequence, when the **IDLE** phase is reached, does the MC_ME enter the **DRUN** mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a **SAFE** mode request issued to the MC_ME or to an interrupt issued to the core (see [Section 7.3.1.3, “Functional Event Reset Disable Register \(RGM_FERD\)”](#) and [Section 7.3.1.4, “Functional Event Alternate Request Register \(RGM_FEAR\)”](#) for ‘functional’ resets).

7.2 External Signal Description

The MC_RGM interfaces to the bidirectional reset pin RESET_B and the boot mode pins FAB, ABS[1:0].

7.3 Memory Map and Register Definition

Table 52. MC_RGM Register Description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FE_4000	RGM_FES	Functional Event Status	half-word	read	read/write ¹	read/write ¹	on page 151
0xC3FE_4002	RGM_DES	Destructive Event Status	half-word	read	read/write ¹	read/write ¹	on page 152
0xC3FE_4004	RGM_FERD	Functional Event Reset Disable	half-word	read	read/write ²	read/write ²	on page 153
0xC3FE_4010	RGM_FEAR	Functional Event Alternate Request	half-word	read	read/write	read/write	on page 155
0xC3FE_4018	RGM_FESS	Functional Event Short Sequence	half-word	read	read/write	read/write	on page 156
0xC3FE_401C	RGM_FBRE	Functional Bidirectional Reset Enable	half-word	read	read/write	read/write	on page 157

¹ individual bits cleared on writing '1'

² write once: '0' = enable, '1' = disable.

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 53. MC_RGM Memory Map

Address	Name	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0xC3FE_4000	RGM_FES / RGM_DES	R	F_EXR						F_FLASH		F_CMU0_FHL	F_CMU0_OLR	F_PLL0	F_CHKSTOP	F_SOFT	F_CORE	F_JTAG	
		W	w1c						w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	
	R	F_POR										F_LVD27_VREG		F_SWT			F_LVD12	
	W	w1c										w1c		w1c			w1c	
0xC3FE_4004	RGM_FERD	R	D_EXR						D_FLASH		D_CMU0_FHL	D_CMU0_OLR	D_PLL0	D_CHKSTOP	D_SOFT	D_CORE	D_JTAG	
		W																
	R	0										D_LVD27_VREG		D_SWT			D_LVD12	
	W																	
0xC3FE_4008 ... 0xC3FE_400C	reserved																	
0xC3FE_4010	RGM_FEAR	R									AR_CMU0_FHL	AR_CMU0_OLR	AR_PLL0				AR_CORE	AR_JTAG
		W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																	
0xC3FE_4014	reserved																	

Table 53. MC_RGM Memory Map (continued)

Address	Name	0		1		2		3		27		5		6		7		8		9		10		11		12		13		14		15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
0xC3FE_4018	RGM_FESS	R	SS_EXR													SS_FLASH					SS_CMU0_FHL	SS_CMU0_OLR		SS_PLLO	SS_CHKSTOP		SS_SOFT		SS_CORE		SS_JTAG		
		W																															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																															
0xC3FE_401C	RGM_FBRE	R	BE_EXR													BE_FLASH					BE_CMU0_FHL	BE_CMU0_OLR		BE_PLLO	BE_CHKSTOP		BE_SOFT		BE_CORE		BE_JTAG		
		W																															
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																															
0xC3FE_4020 ... 0xC3FE_7FFC	reserved																																

7.3.1 Register Descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **RGM_DES[8:15]** register bits may be accessed as a word at address 0xC3FE_4000, as a half-word at address 0xC3FE_4002, or as a byte at address 0xC3FE_4003.

Some fields may be read-only, and their reset value of ‘1’ or ‘0’ and the corresponding behavior cannot be changed.

7.3.1.1 Functional Event Status Register (RGM_FES)

Address 0xC3FE_4000 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_EXR							F_FLASH		F_CMU0_FHL	F_CMU0_OLR	F_PLLO	F_CHKSTOP	F_SOFT	F_CORE	F_JTAG
W	w1c							w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 60. Functional Event Status Register (RGM_FES)

This register contains the status of the last asserted functional reset sources. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1' if the triggering event has already been cleared at the source.

NOTE

If a 'functional' reset source is configured to generate a **SAFE** mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated **RGM_FES** status bit in order to avoid multiple **SAFE** mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the **RGM_FES** has been properly cleared, and if not, clear it again.

Table 54. Functional Event Status Register (RGM_FES) Field Descriptions

Field	Description
F_EXR	Flag for External Reset 0 No external reset event has occurred since either the last clear or the last destructive reset assertion 1 An external reset event has occurred
F_FLASH	Flag for code or data flash fatal error 0 No code or data flash fatal error event has occurred since either the last clear or the last destructive reset assertion 1 A code or data flash fatal error event has occurred
F_CMU0_FHL	Flag for FMPLL_0 clock frequency higher/lower than reference 0 No FMPLL_0 clock frequency higher/lower than reference event has occurred since either the last clear or the last destructive reset assertion 1 A FMPLL_0 clock frequency higher/lower than reference event has occurred
F_CMU0_OLR	Flag for oscillator frequency lower than reference 0 No oscillator frequency lower than reference event has occurred since either the last clear or the last destructive reset assertion 1 A oscillator frequency lower than reference event has occurred

Table 54. Functional Event Status Register (RGM_FES) Field Descriptions (continued)

Field	Description
F_PLL0	Flag for FMPLL_0 fail 0 No FMPLL_0 fail event has occurred since either the last clear or the last destructive reset assertion 1 A FMPLL_0 fail event has occurred
F_CHKSTO P	Flag for checkstop reset 0 No checkstop reset event has occurred since either the last clear or the last destructive reset assertion 1 A checkstop reset event has occurred
F_SOFT	Flag for software reset 0 No software reset event has occurred since either the last clear or the last destructive reset assertion 1 A software reset event has occurred
F_CORE	Flag for core reset 0 No core reset event has occurred since either the last clear or the last destructive reset assertion 1 A core reset event has occurred
F_JTAG	Flag for JTAG initiated reset 0 No JTAG initiated reset event has occurred since either the last clear or the last destructive reset assertion 1 A JTAG initiated reset event has occurred

7.3.1.2 Destructive Event Status Register (RGM_DES)

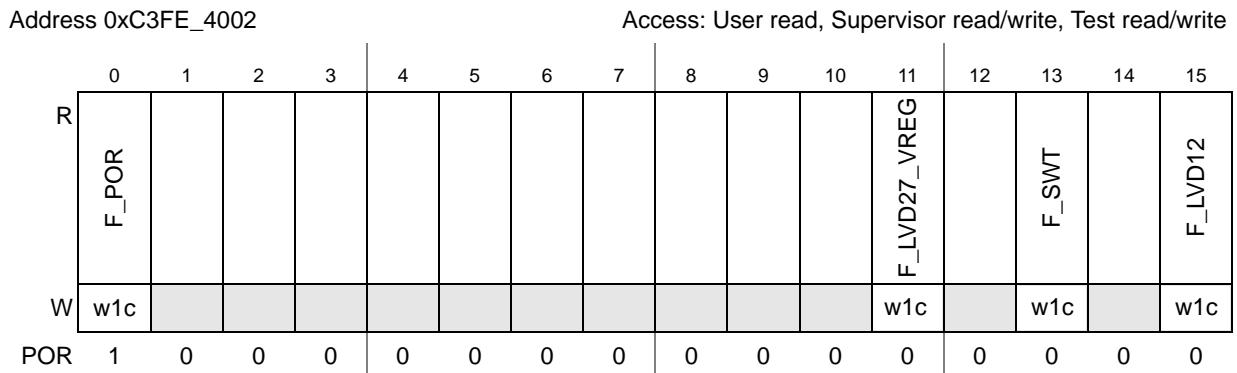


Figure 61. Destructive Event Status Register (RGM_DES)

This register contains the status of the last asserted destructive reset sources. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write ‘1’.

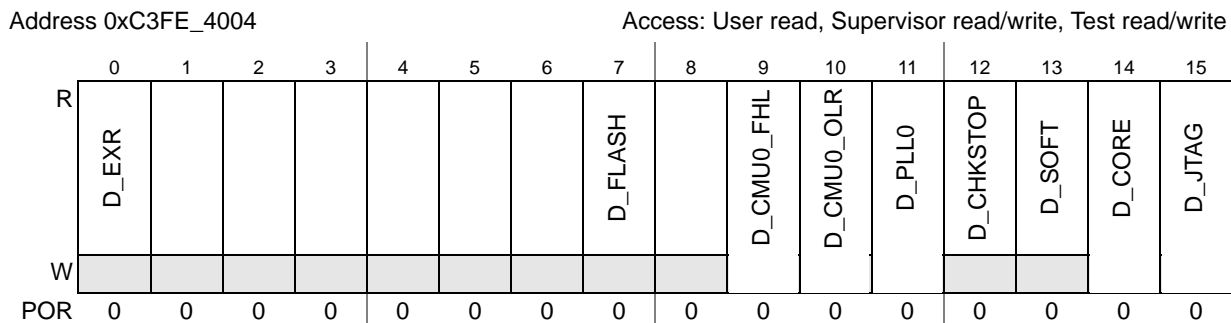
Table 55. Destructive Event Status Register (RGM_DES) Field Descriptions

Field	Description
F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred
F_LVD27_VREG	Flag for 2.7V low-voltage detected (VREG) 0 No 2.7V low-voltage detected (VREG) event has occurred since either the last clear or the last power-on reset assertion 1 A 2.7V low-voltage detected (VREG) event has occurred
F_SWT	Flag for software watchdog timer 0 No software watchdog timer event has occurred since either the last clear or the last power-on reset assertion 1 A software watchdog timer event has occurred
F_LVD12	Flag for 1.2V low-voltage detected 0 No 1.2V low-voltage detected event has occurred since either the last clear or the last power-on reset assertion 1 A 1.2V low-voltage detected event has occurred

NOTE

The **F_POR** flag is automatically cleared on a 1.2 V low-voltage detected or a 2.7 V low-voltage detected. This means that if the power-up sequence is not monotonic (i.e., the voltage rises and then drops enough to trigger a low-voltage detection), the **F_POR** flag may not be set but instead the **F_LVD12** or **F_LVD27_VREG** flag is set on exiting the reset sequence. Therefore, if the **F_POR**, **F_LVD12** or **F_LVD27_VREG** flags are set on reset exit, software should interpret the reset cause as power-on.

7.3.1.3 Functional Event Reset Disable Register (RGM_FERD)


Figure 62. Functional Event Reset Disable Register (RGM_FERD)

This register provides dedicated bits to disable functional reset sources. When a functional reset source is disabled, the associated functional event will trigger either a **SAFE** mode request or an interrupt request (see Section 7.3.1.4, “Functional Event Alternate Request Register (RGM_FEAR)”). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

WARNING

It is important to clear the **RGM_FES** register before setting any of the bits in the **RGM_FERD** register to '1'. Otherwise a redundant **SAFE** mode request or interrupt request may occur.

Table 56. Functional Event Reset Disable Register (RGM_FERD) Field Descriptions

Field	Description
D_EXR	Disable External Reset 0 An external reset event triggers a reset sequence
D_FLASH	Disable code or data flash fatal error 0 A code or data flash fatal error event triggers a reset sequence
D_CMU0_F HL	Disable FMPLL_0 clock frequency higher/lower than reference 0 A FMPLL_0 clock frequency higher/lower than reference event triggers a reset sequence 1 A FMPLL_0 clock frequency higher/lower than reference event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU0_FHL
D_CMU0_O LR	Disable oscillator frequency lower than reference 0 A oscillator frequency lower than reference event triggers a reset sequence 1 A oscillator frequency lower than reference event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU0_OLR
D_PLL0	Disable FMPLL_0 fail 0 A FMPLL_0 fail event triggers a reset sequence 1 A FMPLL_0 fail event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_PLL0
D_CHKSTO P	Disable checkstop reset 0 A checkstop reset event triggers a reset sequence
D_SOFT	Disable software reset 0 A software reset event triggers a reset sequence
D_CORE	Disable core reset 0 A core reset event triggers a reset sequence 1 A core reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CORE
D_JTAG	Disable JTAG initiated reset 0 A JTAG initiated reset event triggers a reset sequence 1 A JTAG initiated reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_JTAG

7.3.1.4 Functional Event Alternate Request Register (RGM_FEAR)

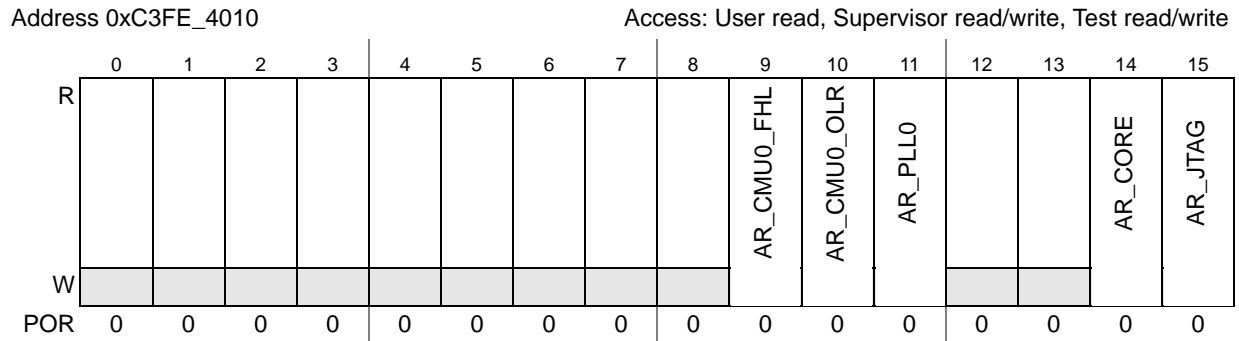


Figure 63. Functional Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a functional event has been disabled. The alternate request can be either a **SAFE** mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Table 57. Functional Event Alternate Request Register (RGM_FEAR) Field Descriptions

Field	Description
AR_CMU0_FHL	Alternate Request for FMPLL_0 clock frequency higher/lower than reference 0 Generate a SAFE mode request on a FMPLL_0 clock frequency higher/lower than reference event if the reset is disabled 1 Generate an interrupt request on a FMPLL_0 clock frequency higher/lower than reference event if the reset is disabled
AR_CMU0_OLR	Alternate Request for oscillator frequency lower than reference 0 Generate a SAFE mode request on a oscillator frequency lower than reference event if the reset is disabled 1 Generate an interrupt request on a oscillator frequency lower than reference event if the reset is disabled
AR_PLL0	Alternate Request for FMPLL_0 fail 0 Generate a SAFE mode request on a FMPLL_0 fail event if the reset is disabled 1 Generate an interrupt request on a FMPLL_0 fail event if the reset is disabled
AR_CORE	Alternate Request for core reset 0 Generate a SAFE mode request on a core reset event if the reset is disabled 1 Generate an interrupt request on a core reset event if the reset is disabled
AR_JTAG	Alternate Request for JTAG initiated reset 0 Generate a SAFE mode request on a JTAG initiated reset event if the reset is disabled 1 Generate an interrupt request on a JTAG initiated reset event if the reset is disabled

7.3.1.5 Functional Event Short Sequence Register (RGM_FESS)

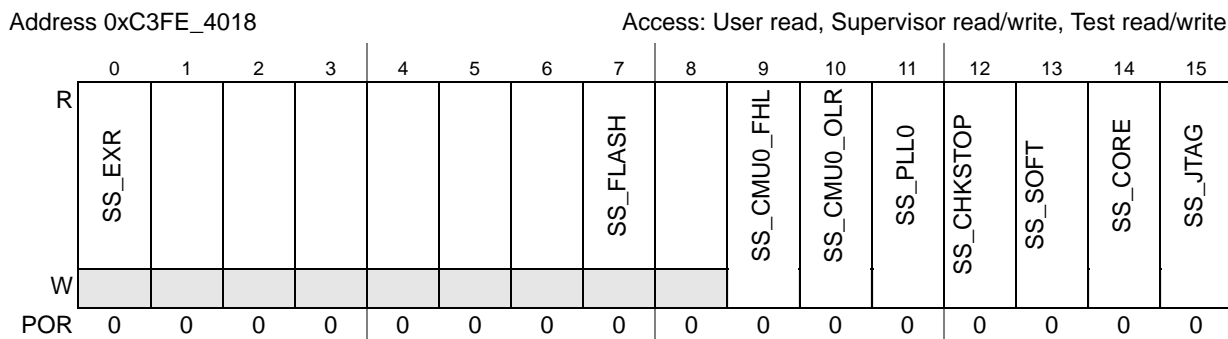


Figure 64. Functional Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence will be done when a functional reset sequence is triggered. The functional reset sequence can either start from **PHASE1** or from **PHASE3**, skipping **PHASE1** and **PHASE2**.

NOTE

This could be useful for fast reset sequence, for example to skip flash reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

Table 58. Functional Event Short Sequence Register (RGM_FESS) Field Descriptions

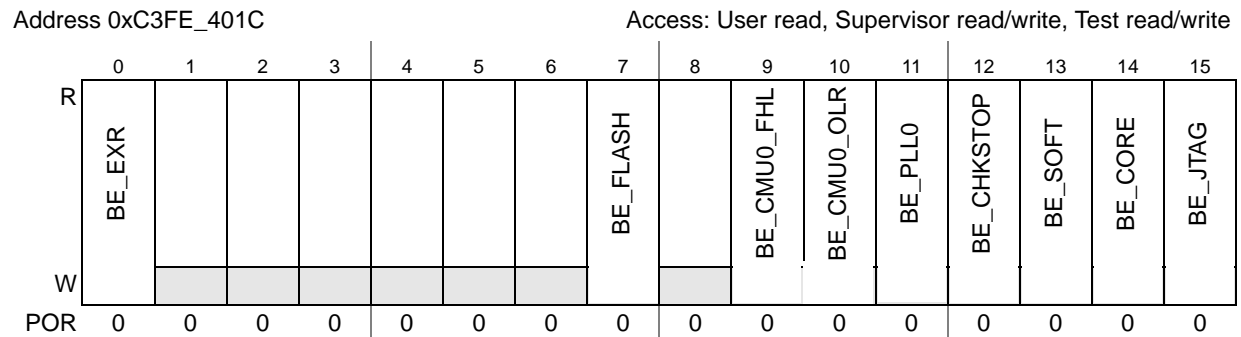
Field	Description
SS_EXR	Short Sequence for External Reset 0 The reset sequence triggered by an external reset event will start from PHASE1
SS_FLASH	Short Sequence for code or data flash fatal error 0 The reset sequence triggered by a code or data flash fatal error event will start from PHASE1
SS_CMU0_FHL	Short Sequence for FMPLL_0 clock frequency higher/lower than reference 0 The reset sequence triggered by a FMPLL_0 clock frequency higher/lower than reference event will start from PHASE1 1 The reset sequence triggered by a FMPLL_0 clock frequency higher/lower than reference event will start from PHASE3 , skipping PHASE1 and PHASE2
SS_CMU0_OLR	Short Sequence for oscillator frequency lower than reference 0 The reset sequence triggered by a oscillator frequency lower than reference event will start from PHASE1 1 The reset sequence triggered by a oscillator frequency lower than reference event will start from PHASE3 , skipping PHASE1 and PHASE2
SS_PLL0	Short Sequence for FMPLL_0 fail 0 The reset sequence triggered by a FMPLL_0 fail event will start from PHASE1 1 The reset sequence triggered by a FMPLL_0 fail event will start from PHASE3 , skipping PHASE1 and PHASE2
SS_CHKSTOP	Short Sequence for checkstop reset 0 The reset sequence triggered by a checkstop reset event will start from PHASE1 1 The reset sequence triggered by a checkstop reset event will start from PHASE3 , skipping PHASE1 and PHASE2

Table 58. Functional Event Short Sequence Register (RGM_FESS) Field Descriptions (continued)

Field	Description
SS_SOFT	Short Sequence for software reset 0 The reset sequence triggered by a software reset event will start from PHASE1 1 The reset sequence triggered by a software reset event will start from PHASE3 , skipping PHASE1 and PHASE2
SS_CORE	Short Sequence for core reset 0 The reset sequence triggered by a core reset event will start from PHASE1 1 The reset sequence triggered by a core reset event will start from PHASE3 , skipping PHASE1 and PHASE2
SS_JTAG	Short Sequence for JTAG initiated reset 0 The reset sequence triggered by a JTAG initiated reset event will start from PHASE1 1 The reset sequence triggered by a JTAG initiated reset event will start from PHASE3 , skipping PHASE1 and PHASE2

NOTE

This register is reset on any enabled ‘destructive’ or ‘functional’ reset event.

7.3.1.6 Functional Bidirectional Reset Enable Register (RGM_FBRE)

Figure 65. Functional Bidirectional Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on functional reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

Table 59. Functional Bidirectional Reset Enable Register (RGM_FBRE) Field Descriptions

Field	Description
BE_EXR	Bidirectional Reset Enable for External Reset 0 RESET_B is asserted on an external reset event if the reset is enabled 1 RESET_B is not asserted on an external reset event
BE_FLASH	Bidirectional Reset Enable for code or data flash fatal error 0 RESET_B is asserted on a code or data flash fatal error event if the reset is enabled 1 RESET_B is not asserted on a code or data flash fatal error event
BE_CMU0_FHL	Bidirectional Reset Enable for FMPLL_0 clock frequency higher/lower than reference 0 RESET_B is asserted on a FMPLL_0 clock frequency higher/lower than reference event if the reset is enabled 1 RESET_B is not asserted on a FMPLL_0 clock frequency higher/lower than reference event

Table 59. Functional Bidirectional Reset Enable Register (RGM_FBRE) Field Descriptions (continued)

Field	Description
BE_CMU0_OLR	Bidirectional Reset Enable for oscillator frequency lower than reference 0 RESET_B is asserted on a oscillator frequency lower than reference event if the reset is enabled 1 RESET_B is not asserted on a oscillator frequency lower than reference event
BE_PLL0	Bidirectional Reset Enable for FMPLL_0 fail 0 RESET_B is asserted on a FMPLL_0 fail event if the reset is enabled 1 RESET_B is not asserted on a FMPLL_0 fail event
BE_CHKSTOP	Bidirectional Reset Enable for checkstop reset 0 RESET_B is asserted on a checkstop reset event if the reset is enabled 1 RESET_B is not asserted on a checkstop reset event
BE_SOFT	Bidirectional Reset Enable for software reset 0 RESET_B is asserted on a software reset event if the reset is enabled 1 RESET_B is not asserted on a software reset event
BE_CORE	Bidirectional Reset Enable for core reset 0 RESET_B is asserted on a core reset event if the reset is enabled 1 RESET_B is not asserted on a core reset event
BE_JTAG	Bidirectional Reset Enable for JTAG initiated reset 0 RESET_B is asserted on a JTAG initiated reset event if the reset is enabled 1 RESET_B is not asserted on a JTAG initiated reset event

7.4 Functional Description

7.4.1 Reset State Machine

The main role of MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in [Table 60](#).

Table 60. MC_RGM Reset Implications

Source	What Gets Reset	External Reset Assertion ¹	Boot Mode Capture
power-on reset	all	yes	yes
'destructive' resets	all except some clock/reset management	yes	yes
external reset	all except some clock/reset management and debug	programmable ²	yes
'functional' resets	all except some clock/reset management and debug	programmable ²	programmable ³
shortened 'functional' resets ⁴	flip-flops except some clock/reset management	programmable ²	programmable ³

¹ 'external reset assertion' means that the RESET_B pin is asserted by the MC_RGM until the end of reset **PHASE3**

² the assertion of the external reset is controlled via the **RGM_FBRE** register

³ the boot mode is captured if the external reset is asserted

⁴ the short sequence is enabled via the **RGM_FESS** register

NOTE

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in [Figure 66](#).

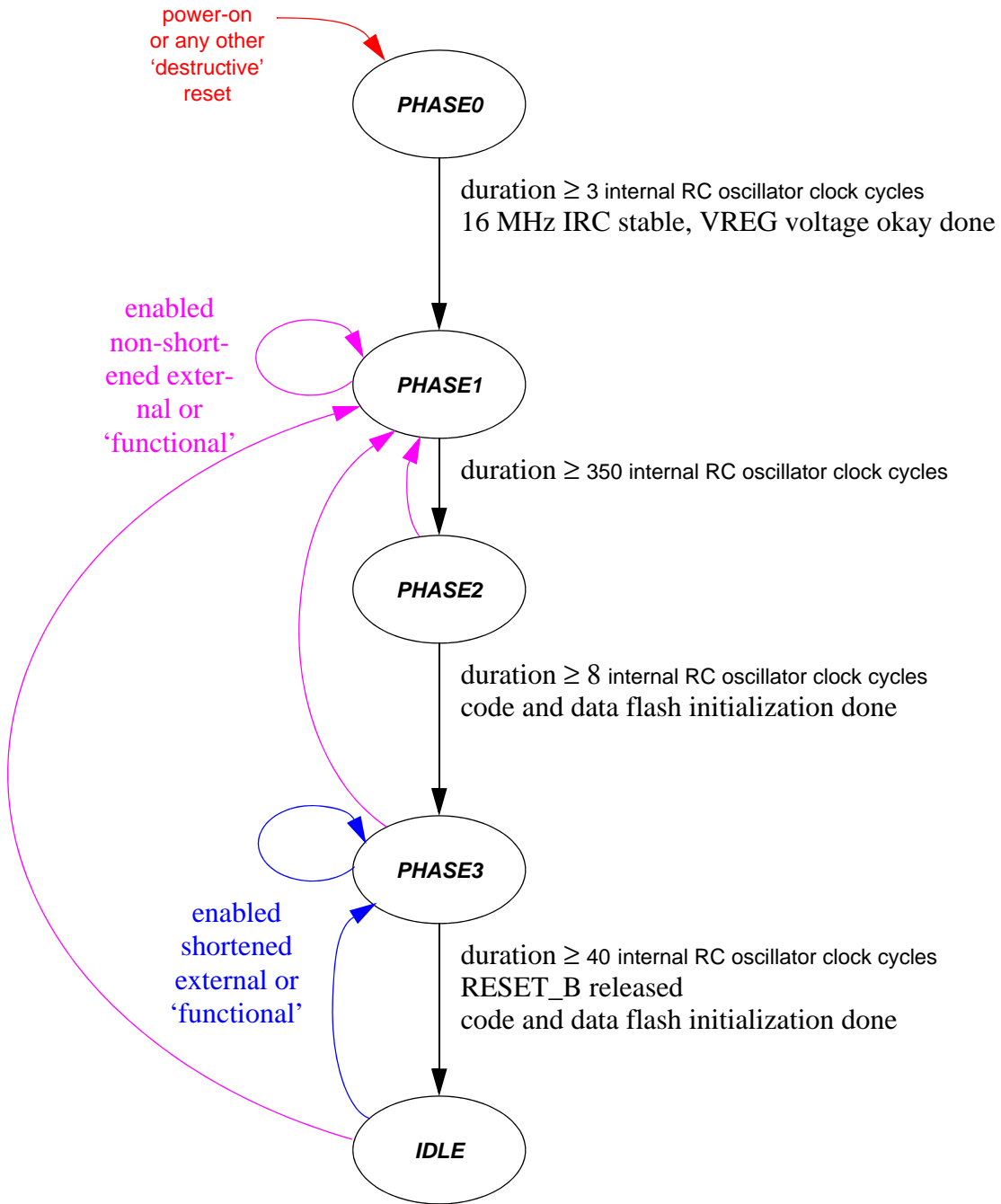


Figure 66. MC_RGM State Machine

7.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or any other 'destructive' reset event. The reset state machine exits **PHASE0** and enters **PHASE1** on verification of the following:

- all enabled 'destructive' resets have been processed
- all processes that need to be done in **PHASE0** are completed

- 16 MHz IRC stable, VREG voltage okay
- a minimum of 3 internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled ‘destructive’ reset event

7.4.1.2 PHASE1 Phase

This phase is entered either on exit from **PHASE0** or immediately from **PHASE2**, **PHASE3**, or **IDLE** on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits **PHASE1** and enters **PHASE2** on verification of the following:

- all enabled, non-shortened ‘functional’ resets have been processed
- a minimum of 350 internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

7.4.1.3 PHASE2 Phase

This phase is entered on exit from **PHASE1**. The reset state machine exits **PHASE2** and enters **PHASE3** on verification of the following:

- all processes that need to be done in **PHASE2** are completed
 - code and data flash initialization
- a minimum of 8 internal RC oscillator clock cycles have elapsed since entering **PHASE2**

7.4.1.4 PHASE3 Phase

This phase is entered either on exit from **PHASE2** or immediately from **IDLE** on an enabled, shortened ‘functional’ reset event. The reset state machine exits **PHASE3** and enters **IDLE** on verification of the following:

- all processes that need to be done in **PHASE3** are completed
 - code and data flash initialization
- a minimum of 40 internal RC oscillator clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event

7.4.1.5 IDLE Phase

This is the final phase and is entered on exit from **PHASE3**. When this phase is reached, the MC_RGM releases control of the chip to the core and waits for new reset events that can trigger a reset sequence.

7.4.2 Destructive Resets

A ‘destructive’ reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given ‘destructive’ reset event (**RGM_DES.F_<destructive reset>** bit) is set when the ‘destructive’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

The chip's low-voltage detector threshold ensures that, when 1.2V low-voltage detected, the supply is sufficient to have the destructive event correctly propagated through the digital logic. Therefore, if a given 'destructive' reset is asserted, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. A destructive reset will trigger a reset sequence starting from the beginning of **PHASE0**.

7.4.3 External Reset

The MC_RGM manages the external reset coming from RESET_B. The detection of a falling edge on RESET_B will start the reset sequence from the beginning of **PHASE1**.

The status flag associated with the external reset falling edge event (**RGM_FES.F_EXR** bit) is set when the external reset is asserted and the power-on reset is not asserted.

NOTE

The **RGM_FERD** register can be written only once between two power-on reset events.

External reset will trigger a reset sequence starting from the beginning of **PHASE1**.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event
- an external reset event
- a 'functional' reset event configured via the **RGM_FBRE** register to assert the external reset

In this case, the external reset is asserted until the end of **PHASE3**.

7.4.4 Functional Resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (**RGM_FES.F_<functional reset>** bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

The 'functional' reset can be optionally disabled by software writing bit **RGM_FERD.D_<functional reset>**.

NOTE

The **RGM_FERD** register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of **PHASE1**. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence

triggered by a functional reset. When **RGM_FESS.SS_<functional reset>** is set, the associated ‘functional’ reset will trigger a reset sequence starting directly from the beginning of **PHASE3**, skipping **PHASE1** and **PHASE2**. This can be useful especially in case a functional reset should not reset the flash module.

7.4.5 Alternate Event Generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a **SAFE** mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the **RGM_FERD** and **RGM_FEAR** registers as shown in [Table 61](#).

Table 61. MC_RGM Alternate Event Selection

RGM_FERD Bit Value	RGM_FEAR Bit Value	Generated Event
0	X	reset
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (i.e., at the reset source that caused the alternate request) and also clearing the appropriate **RGM_FES** status bit.

NOTE

Alternate requests (**SAFE** mode as well as interrupt requests) are generated regardless of whether the system clock is running.

NOTE

If a masked ‘functional’ reset event which is configured to generate a **SAFE** mode/interrupt request occurs during **PHASE1**, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

7.4.6 Boot Mode Capturing

The MC_RGM samples FAB, ABS[1:0] whenever RESET_B is asserted until five internal RC oscillator clock cycles before its deassertion edge. The result of the sampling is used at the beginning of reset **PHASE3** for boot mode selection and is retained after RESET_B has been deasserted for subsequent boots after reset sequences during which RESET_B is not asserted.

NOTE

In order to ensure that the boot mode is correctly captured, the application needs to apply the valid boot mode value the entire time that RESET_B is asserted.

NOTE

RESET_B can be asserted as a consequence of the internal reset generation. This will force re-sampling of the boot mode pins. (See [Table 60](#) for details.)

Chapter 8

Power Control Unit (MC_PCU)

8.1 Introduction

8.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMU peripheral to the MC_PCU address space.

[Figure 67](#) depicts the MC_PCU block diagram.

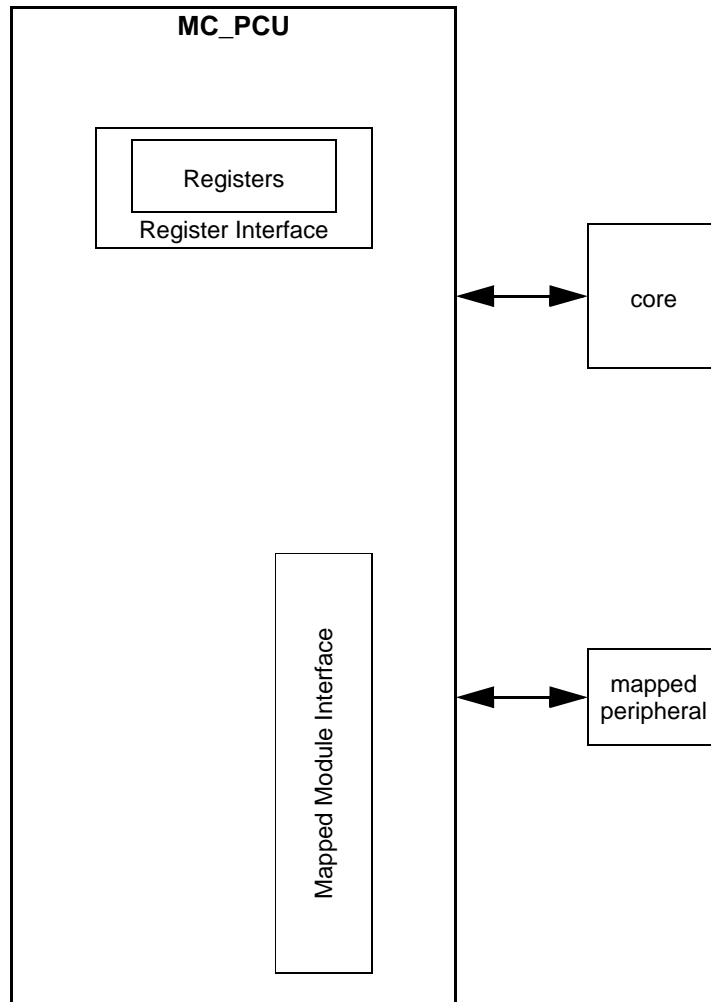


Figure 67. MC_PCU Block Diagram

8.1.2 Features

The MC_PCU includes the following features:

- maps the PMU registers to the MC_PCU address space

8.2 External Signal Description

The MC_PCU has no connections to any external pins.

8.3 Memory Map and Register Definition

8.3.1 Memory Map

Table 62. MC_PCU Register Description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FE_8040	PCU_PSTAT	Power Domain Status Register	word	read	read	read	on page 168

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 63. MC_PCU Memory Map

Address	Name	Bit																
		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	
0xC3FE_8004 ... 0xC3FE_803C		reserved																
0xC3FE_8040	PCU_PSTAT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		W																
		R																PDO
		W																
0x044 ... 0x07C		reserved																
0xC3FE_8080 ... 0xC3FE_80FC		PMU registers																
0xC3FE_8100 ... 0xC3FE_BFFC		reserved																

8.3.2 Register Descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **PDO** field of the **PCU_PSTAT** register may be accessed as a word at address 0xC3FE_8040, as a half-word at address 0xC3FE_8042, or as a byte at address 0xC3FE_8043.

8.3.2.1 Power Domain Status Register (PCU_PSTAT)

Address 0xC3FE_8040				Access: User read, Supervisor read, Test read												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																PDO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 68. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 64. Power Domain Status Register (PCU_PSTAT) Field Descriptions

Field	Description
PD n	Power status for power domain # n 0 Power domain is inoperable 1 Power domain is operable

Chapter 9 Power Management

9.1 Power management overview

The device supports the following power modes:

- Internal voltage regulation mode
- External voltage regulation mode

9.1.1 Internal voltage regulation mode

In this mode, the following supplies are involved:

- $V_{DD_HV_IO}$ (3.3V) — This is the main supply provided externally.
- $V_{DD_LV_CORE}$ (1.2 V) — This is the core logic supply. In the internal regulation mode, the core supply is derived from the main supply via an on-chip linear regulator driving an internal PMOS ballast transistor. The PMOS ballast transistors are located in the pad ring and their source connectors are directly bonded to a dedicated pin. See [Figure 69](#).

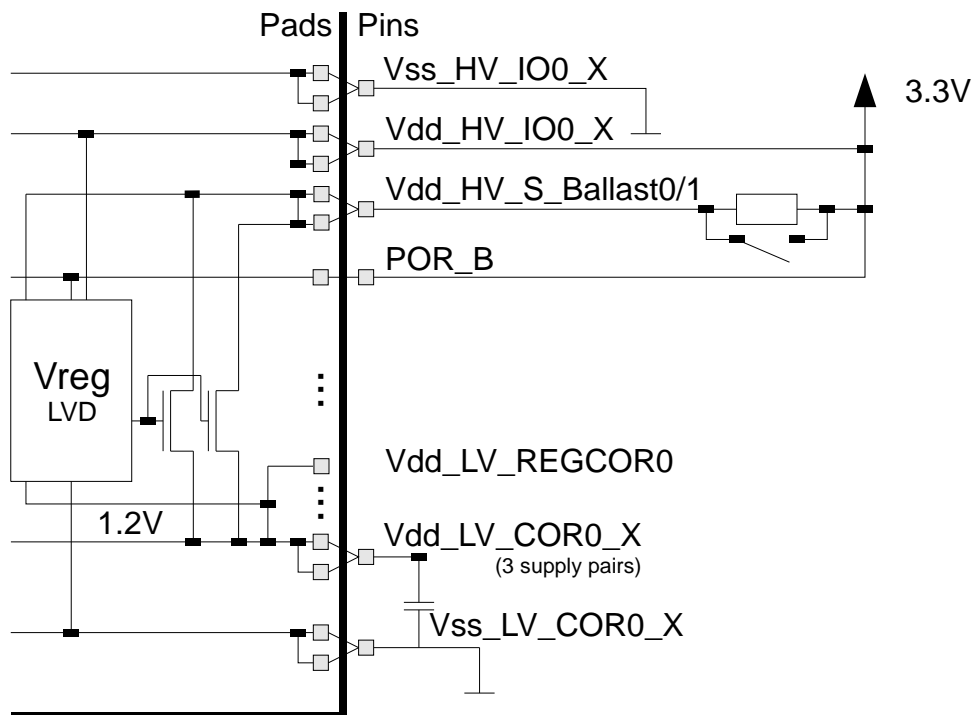


Figure 69. Internal Regulation Mode

The core supply can also be provided externally. Table 65 shows how to connect $V_{DD_HV_S_BALLAST}$ pin for internal and external core supply mode.

Table 65. Core Supply Select

Mode	Vdd_S_Ballast
Internal supply mode (via internal PMOS ballast transistors)	$V_{DD_HV_IO}$ (3.3V)
External supply mode (e.g., via external switched regulator)	$V_{DD_LV_CORE}$ (1.2V)

9.1.2 External voltage regulation mode

In the external regulation mode, the core supply is provided externally using a switched regulator. This saves on-chip power consumption by avoiding the voltage drop over the ballast transistor. The external supply mode is selected via a board level supply change at the $V_{dd_HV_S_Ballast}$ pin.

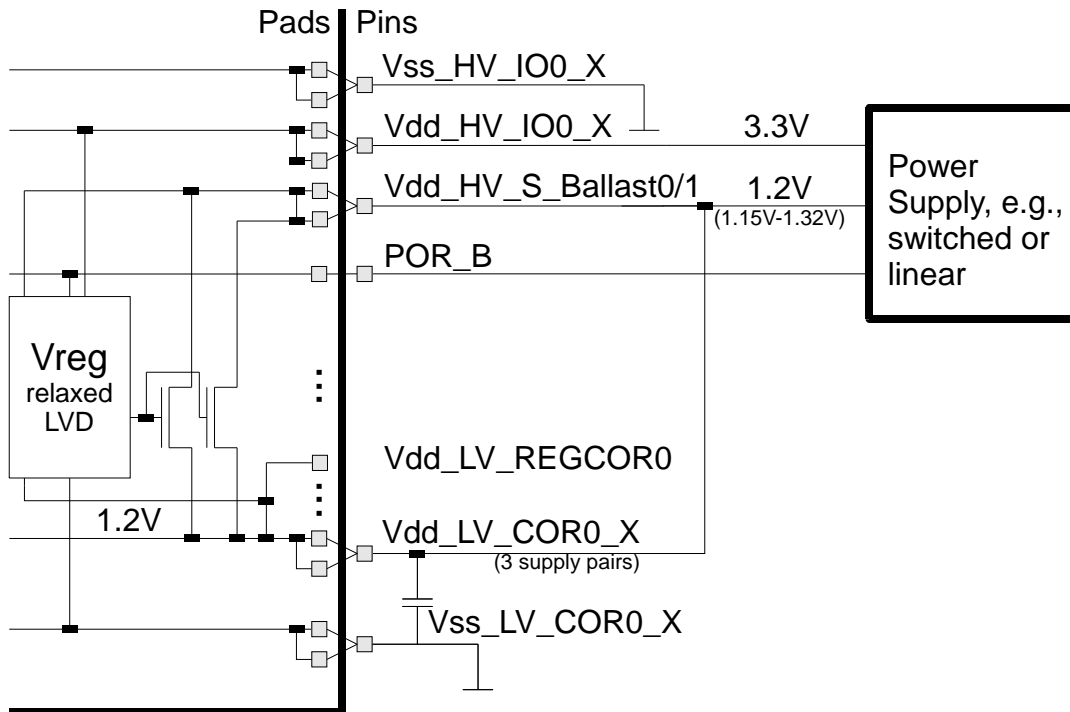


Figure 70. External Regulation Mode

9.1.3 Voltage Regulator Electrical Characteristics

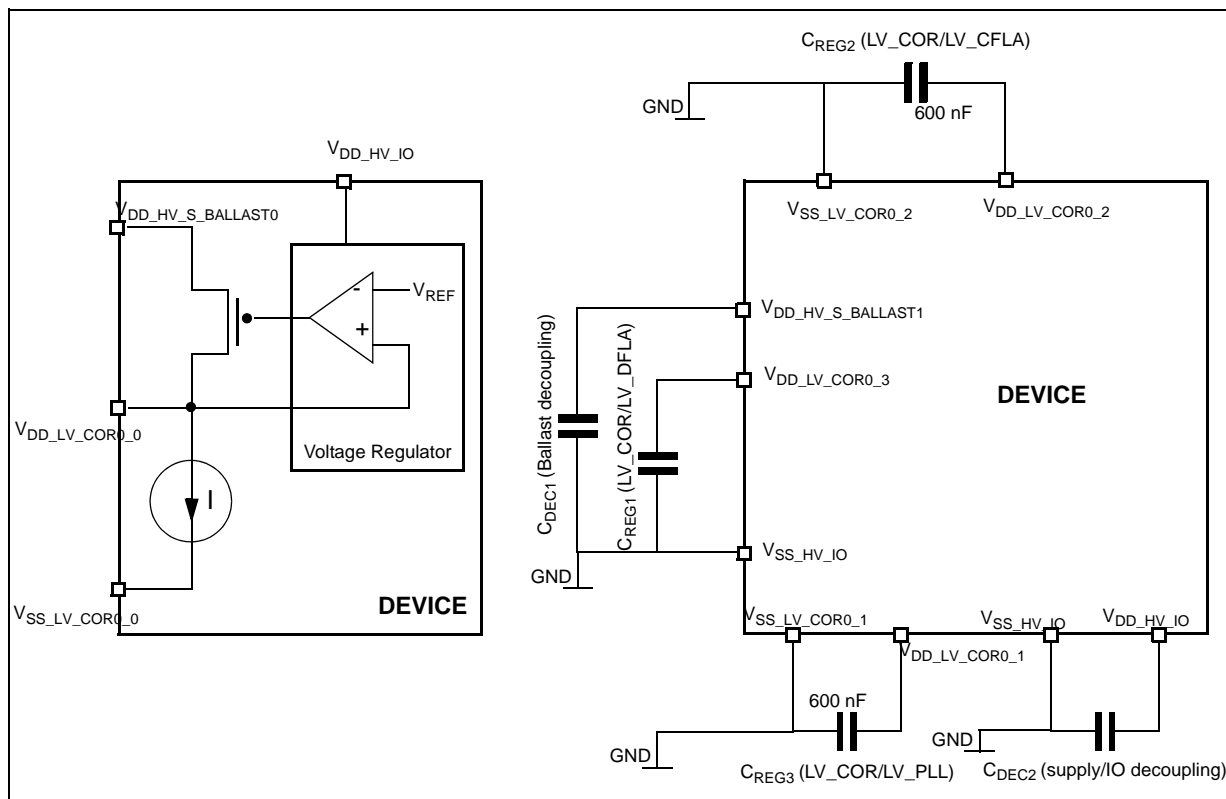


Figure 71. Voltage regulator capacitance connection

9.2 Power sequencing

As shown in [Figure 72](#) the MPC5604E includes on-chip diodes for ESD protection.

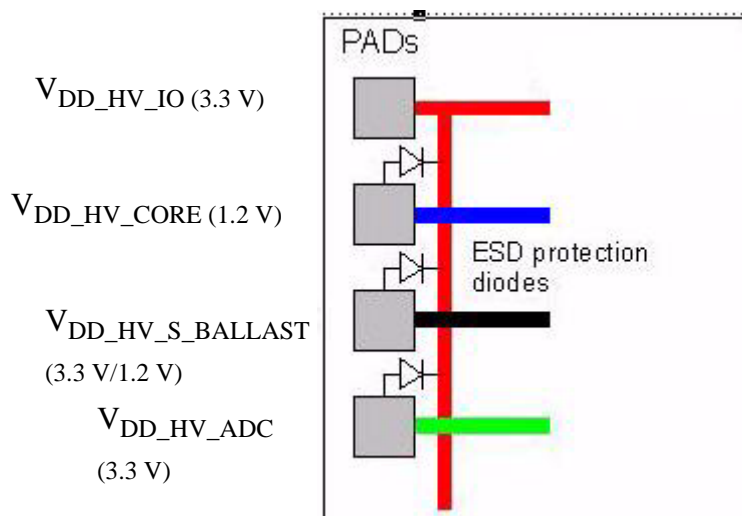


Figure 72. Internal diodes between supply pads

For both internal and external core voltage supply modes the following conditions must be guaranteed at power up and power down:

- $V_{DD_HV_IO} \geq V_{DD_LV_CORE}$
- $V_{DD_HV_IO} \geq V_{DD_HV_S_BALLAST}$
- $V_{DD_HV_IO} \geq V_{DD_HV_ADC}$

9.3 Power Management Unit (PMU)

The primary function of the MPC5604E's power management unit (PMU) is to generate the 1.2 V core logic supply from the 3.3 V main supply. To allow an easy integration into a system The PMU includes an internal PMOS ballast transistor.

In addition the PMU monitors the operation voltages using a set of supervisory circuits: the LVDs (Low Voltage Detectors). Furthermore, the Power On Reset (POR) circuit monitors VddREG, the voltage used internally by the PMU. When VddREG is below the POR threshold voltage, the POR output is asserted, otherwise it is deasserted.

The purpose of the POR circuit is to keep the MPC5604E in the reset state as long as the supply voltage to the LVD circuits is below their minimum operating voltage. By the time the POR output deasserts, the LVDs are operating and able to assert their outputs properly.

In summary the PMU has the following features:

- Internal PMOS ballast transistor
- Power On Reset (POR)
- Low voltage detection
- Internal power on reset (POR) circuit to detect minimal voltage to operate voltage regulator.
- LVD27 for VddIO. The minimum threshold value must not be below 2.60V to guarantee correct flash memory read operations.
- LVD12 for VddCore (trimmed during testing)

Chapter 10

Interrupt Controller (INTC)

10.1 Introduction

The INTC provides priority-based preemptive scheduling of interrupt service requests (ISRs). This scheduling scheme is suitable for statically scheduled hard real-time systems. The INTC supports 106 interrupt requests. It is targeted to work with Power Architecture technology and automotive applications where the ISRs nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, the time from the assertion of the peripheral's interrupt request to when the processor is performing useful work to service the interrupt request needs to be minimized. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 16 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Because each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that tasks sharing the resource will not preempt each other.

Multiple processors can assert interrupt requests to each other through software configurable interrupt requests. These software configurable interrupt requests can also be used to separate the work involved in servicing an interrupt request into a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software configurable interrupt request to finish the servicing in a lower priority ISR. Therefore these software configurable interrupt requests can be used instead of the peripheral ISR scheduling a task through the RTOS.

10.2 Features

- Supports 106 peripheral interrupts and 8 software-configurable interrupt request sources
- Unique 9-bit vector per interrupt source
- Each interrupt source programmable to one of 16 priorities
- Preemption
 - Preemptive prioritized interrupt requests to processor
 - ISR at a higher priority preempts ISRs or tasks at lower priorities
 - Automatic pushing or popping of preempted priority to or from a LIFO
 - Ability to modify the ISR or task priority; modifying the priority can be used to implement the priority ceiling protocol for accessing shared resources.

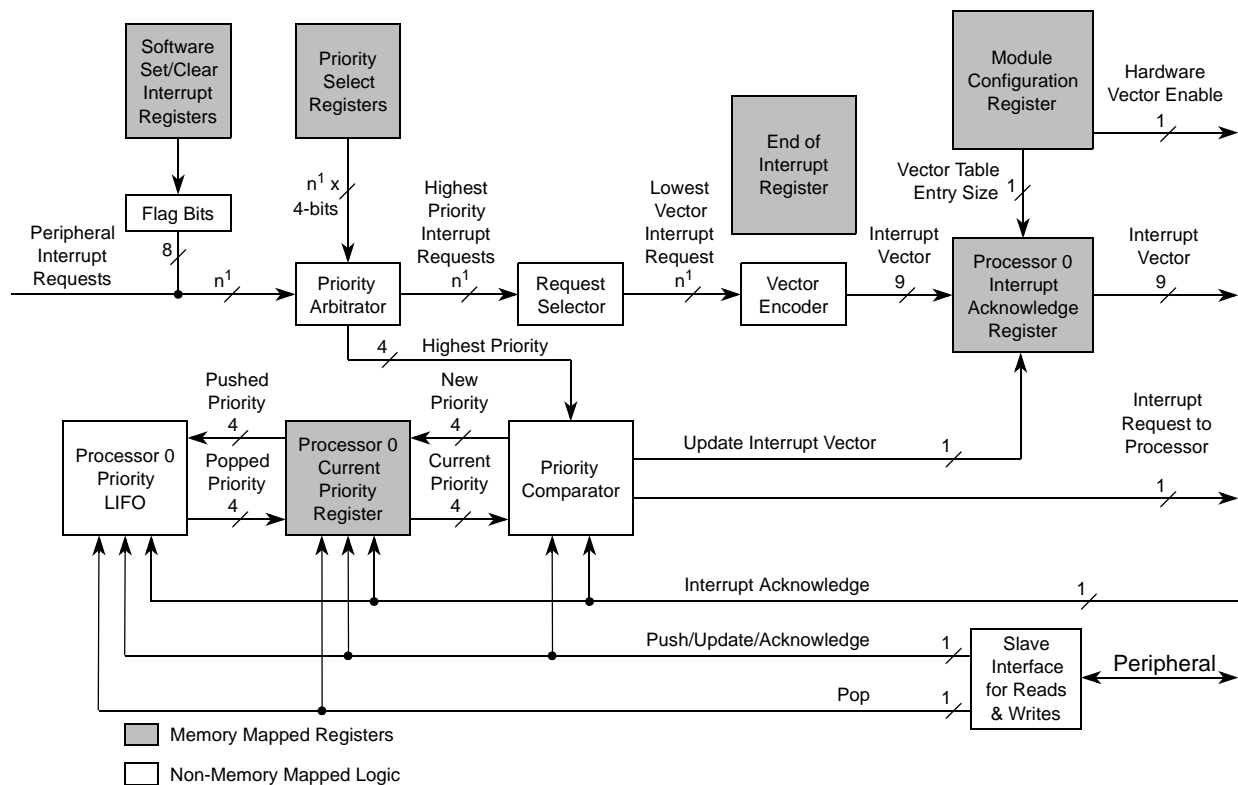
- Low latency—3 clock cycles from receipt of interrupt request from peripheral to interrupt request to processor

Table 66. Interrupt sources available

Interrupt sources (106)	Number available
Software	8
eDMA2x	17
SWT	1
STM	4
SIUL	4
MC_ME	4
MC_RGM	1
MCM	3
I ² C	2
Video Encoder	6
Fast Ethernet Controller (FEC)	3
CE_RTC	1
XOSC	1
PTP	1
SAI	6
PIT	4
ADC	3
FlexCAN	8
eTimer	8
DSPI	15
LINFlex	6

10.3 Block diagram

Figure 73 shows a block diagram of the interrupt controller (INTC).



¹ The total number of available interrupt sources is 106, which includes 8 software sources.

Figure 73. INTC block diagram

10.4 Modes of operation

10.4.1 Normal mode

In normal mode, the INTC has two handshaking modes with the processor: software vector mode and hardware vector mode.

NOTE

To correctly configure the interrupts in both software and hardware vector mode, the user must also configure the IVPR. The core register IVPR contains the base address for the interrupt handlers. Please refer to the core reference manual for more information.

10.4.1.1 Software vector mode

In software vector mode, the interrupt exception handler software must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN bit in INTC_MCR is negated. The hardware vector enable signal to processor 0 or processor 1 is driven as negated when its associated HVEN bit is negated. The vector is read from INC_IACKR. Reading the INTC_IACKR negates the interrupt request to the

associated processor. Even if a higher priority interrupt request arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in INTC_CPR onto the associated LIFO and updates PRI in the associated INTC_CPR with the new priority.

Furthermore, the interrupt vector to the processor is driven as all 0s. The interrupt acknowledge signal from the associated processor is ignored.

10.4.1.2 Hardware vector mode

In hardware vector mode, the hardware signals the interrupt vector from the INTC in conjunction with a processor that can use that vector. This hardware causes the first instruction to be executed in handling the interrupt request to the processor to be specific to that vector. Therefore, the interrupt exception handler is specific to a peripheral or software configurable interrupt request rather than being common to all of them. The INTC uses hardware vector mode for a given processor when the associated HVEN bit in the INTC_MCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted. When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software configurable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR field in the INTC_IACKR, depending on which processor was assigned to handle a given interrupt source.

The processor negates the interrupt request to the processor driven by the INTC by asserting the interrupt acknowledge signal for one clock. Even if a higher priority interrupt request arrived while waiting for the interrupt acknowledge, the interrupt request to the processor will negate for at least one clock.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR does not occur when the associated interrupt acknowledge signal asserts and INTC_SSCIR0_3–INTC_SSCIR4_7 is written at a time such that the PRI value in the associated INTC_CPR register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR is updated with the new priority, and the associated LIFO is neither pushed or popped.

10.4.1.3 Debug mode

The INTC operation in debug mode is identical to its operation in normal mode.

10.4.1.4 Stop mode

The INTC supports stop mode. The INTC can have its clock input disabled at any time by the clock driver on the device. While its clocks are disabled, the INTC registers are not accessible.

The INTC requires clocking in order for a peripheral interrupt request to generate an interrupt request to the processor.

10.5 Memory map and registers description

10.5.1 Module memory map

Table 67 shows the INTC memory map.

Table 67. INTC memory map

Offset from INTC_BASE (0xFFFF4_8000)	Register	Access	Reset value	Location
0x0000	INTC Module Configuration Register (INTC_MCR)	R/W	0x0000_0000	on page 178
0x0004	Reserved			
0x0008	INTC Current Priority Register for Processor (INTC_CPR)	R/W	0x0000_000F	on page 178
0x000C	Reserved			
0x0010	INTC Interrupt Acknowledge Register (INTC_IACKR)	R ¹ /W	0x0000_0000	on page 180
0x0014	Reserved			
0x0018	INTC End-of-Interrupt Register (INTC_EOIR)	W	0x0000_0000	on page 180
0x001C	Reserved			
0x0020–0x0027	INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7)	R/W	0x0000_0000	on page 181
0x0028–0x003C	Reserved			
0x0040–0x011C	INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR220_221) ²	R/W	0x0000_0000	on page 182
0x0120–0x3FFF	Reserved			

¹ When the HVEN bit in the INTC module configuration register (INTC_MCR) is asserted, a read of the INTC_IACKR has no side effects.

² The PRI fields are “reserved” for peripheral interrupt requests whose vectors are labeled as Reserved in Table 74.

10.5.2 Registers description

With exception of the INTC_SSCIR_n and INTC_PSR_n, all registers are 32 bits in width. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle 2 bytes, and aligned 32 bits.

Although INTC_SSCIR_n and INTC_PSR_n are 8 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

In software vector mode, the side effects of a read of INTC_IACKR are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to either INTC_SSCIR0_3–INTC_SSCIR4_7 or INTC_EOIR does not affect the operation of the write.

INTC registers are accessible only when the core is in supervisor mode.

10.5.2.1 INTC Module Configuration Register (INTC_MCR)

The module configuration register configures options of the INTC.

Address: Base + 0x0000 Access: User read/write

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

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

Figure 74. INTC Module Configuration Register (INTC_MCR)

Table 68. INTC_MCR field descriptions

Field	Description
26 VTES	Vector table entry size Controls the number of 0s to the right of INTVEC in Section 10.5.2.3, “INTC Interrupt Acknowledge Register (INTC_IACKR)” . If the contents of INTC_IACKR are used as an address of an entry in a vector table as in software vector mode, then the number of right most 0s will determine the size of each vector table entry. VTES impacts software vector mode operation but also affects INTC_IACKR[INTVEC] position in both hardware vector mode and software vector mode. 0 4 bytes 1 8 bytes
31 HVEN	Hardware vector enable Controls whether the INTC is in hardware vector mode or software vector mode. Refer to Section 10.4, “Modes of operation” , for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

10.5.2.2 INTC Current Priority Register for Processor (INTC_CPR)

Address: Base + 0x0008 Access: User read/write

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

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

Figure 75. INTC Current Priority Register (INTC_CPR)

Table 69. INTC_CPR field descriptions

Field	Description
28–31 PRI[]	Priority PRI is the priority of the currently executing ISR according to the following: 1111 Priority 15—highest priority 1110 Priority 14 1101 Priority 13 1100 Priority 12 1011 Priority 11 1010 Priority 10 1001 Priority 9 1000 Priority 8 0111 Priority 7 0110 Priority 6 0101 Priority 5 0100 Priority 4 0011 Priority 3 0010 Priority 2 0001 Priority 1 0000 Priority 0—lowest priority

The INTC_CPR masks any peripheral or software settable interrupt request set at the same or lower priority as the current value of the INTC_CPR[PRI] field from generating an interrupt request to the processor. When the INTC interrupt acknowledge register (INTC_IACKR) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR) is written, the LIFO is popped into the INTC_CPR's PRI field.

The masking priority can be raised or lowered by writing to the PRI field, supporting the PCP. Refer to [Section 10.7.5, “Priority ceiling protocol”](#).

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource can result in a non-coherent access to the resource. Refer to [Section 10.7.5.2, “Ensuring coherency”](#), for example code to ensure coherency.

10.5.2.3 INTC Interrupt Acknowledge Register (INTC_IACKR)

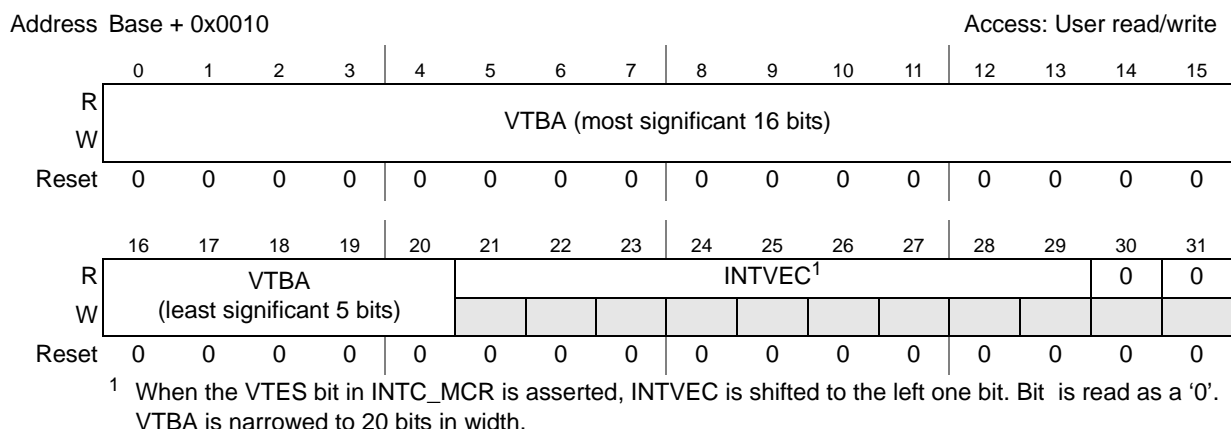


Figure 76. INTC Interrupt Acknowledge Register (INTC_IACKR)

Table 70. INTC_IACKR field descriptions

Field	Description
0–20 or 0–19 VTBA	Vector Table Base Address Can be the base address of a vector table of addresses of ISRs. The VTBA only uses the leftmost 20 bits when the VTES bit in INTC_MCR is asserted.
21–29 or 20–28 INTVEC	Interrupt Vector It is the vector of the peripheral or software configurable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode. Note: If INTC_MCR[VTES] = 1, then the INTVEC field is shifted left one position to bits 20–28. VTBA is then shortened by one bit to bits 0–19.

The interrupt acknowledge register provides a value that can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

In software vector mode, the INTC_IACKR has side effects from reads. Therefore, it must not be speculatively read while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR does not have side effects in hardware vector mode.

10.5.2.4 INTC End-of-Interrupt Register (INTC_EOIR)

Writing to the end-of-interrupt register signals the end of the servicing of the interrupt request. When the INTC_EOIR is written, the priority last pushed on the LIFO is popped into INTC_CPR. An exception to this behavior is described in [Section 10.4.1.2, “Hardware vector mode”](#). The values and size of data written to the INTC_EOIR are ignored. The values and sizes written to this register neither update the INTC_EOIR contents or affect whether the LIFO pops. For possible future compatibility, write four bytes of all 0s to the INTC_EOIR.

Reading the INTC_EOIR has no effect on the LIFO.

Offsets: Base + 0x0018 (IINTC_EOIR) Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	EOI[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	EOI[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 77. INTC End of Interrupt Register for Processor (INTC_EOIR)

Table 71. INTC_EOIR field descriptions

Field	Description
EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

10.5.2.5 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7)

Address Base + 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	CLR	0	0	0	0	0	0	0	CLR
W							SET0	0							SET1	1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CLR	0	0	0	0	0	0	0	CLR
W							SET2	2							SET3	3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 78. INTC Software Set/Clear Interrupt Register 0–3 (INTC_SSCIR[0:3])

Address Base + 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	CLR	0	0	0	0	0	0	0	CLR
W							SET4	4							SET5	5
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CLR	0	0	0	0	0	0	0	CLR
W							SET6	6							SET7	7
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 79. INTC Software Set/Clear Interrupt Register 4–7 (INTC_SSCIR[4:7])

Table 72. INTC_SSCIR[0:7] field descriptions

Field	Description
6, 14, 22, 30 SET[0:7]	Set Flag Bits Writing a '1' sets the corresponding CLR _x bit. Writing a '0' has no effect. Each SET _x always will be read as a '0'.
7, 15, 23, 31 CLR[0:7]	Clear Flag Bits CLR _x is the flag bit. Writing a '1' to CLR _x clears it provided that a '1' is not written simultaneously to its corresponding SET _x bit. Writing a '0' to CLR _x has no effect. 0 Interrupt request not pending within INTC 1 Interrupt request pending within INTC

The software set/clear interrupt registers support the setting or clearing of software configurable interrupt request. These registers contain eight independent sets of bits to set and clear a corresponding flag bit by software. Excepting being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC like a peripheral interrupt request. Writing a '1' to SET_x will leave SET_x unchanged at 0 but sets CLR_x. Writing a '0' to SET_x has no effect. CLR_x is the flag bit. Writing a '1' to CLR_x clears it. Writing a '0' to CLR_x has no effect. If a '1' is written simultaneously to a pair of SET_x and CLR_x bits, CLR_x will be asserted, regardless of whether CLR_x was asserted before the write.

10.5.2.6 INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR220_221)

Address Base + 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PRI0				0	0	0	0	PRI1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRI2				0	0	0	0	PRI3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 80. INTC Priority Select Register 0–3 (INTC_PSR[0:3])

Address Base + 0x011C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PRI220				0	0	0	0	PRI221			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 81. INTC Priority Select Register220–221 (INTC_PSR[220:221])

Table 73. INTC_PSR0_3–INTC_PSR220–221 field descriptions

Field	Description
4–7, 12–15, 20–23, 28–31 PRI[]– PRI220:221	Priority Select PRIx selects the priority for interrupt requests. Refer to Section 10.6, “Functional description” .

Table 74. INTC Priority Select Register address offsets

INTC_PSRx_x	Offset Address	INTC_PSRx_x	Offset Address
INTC_PSR0_3	0x0040	INTC_PSR112_115	0x00B0
INTC_PSR4_7	0x0044	INTC_PSR116_119	0x00B4
INTC_PSR8_11	0x0048	INTC_PSR120_123	0x00B8
INTC_PSR12_15	0x004C	INTC_PSR124_127	0x00BC
INTC_PSR16_19	0x0050	INTC_PSR128_131	0x00C0
INTC_PSR20_23	0x0054	INTC_PSR132_135	0x00C4
INTC_PSR24_27	0x0058	INTC_PSR136_139	0x00C8
INTC_PSR28_31	0x005C	INTC_PSR140_143	0x00CC
INTC_PSR32_35	0x0060	INTC_PSR144_147	0x00D0
INTC_PSR36_39	0x0064	INTC_PSR148_151	0x00D4
INTC_PSR40_43	0x0068	INTC_PSR152_155	0x00D8
INTC_PSR44_47	0x006C	INTC_PSR156_159	0x00DC
INTC_PSR48_51	0x0070	INTC_PSR160_163	0x00E0
INTC_PSR52_55	0x0074	INTC_PSR164_167	0x00E4
INTC_PSR56_59	0x0078	INTC_PSR168_171	0x00E8
INTC_PSR60_63	0x007C	INTC_PSR172_175	0x00EC
INTC_PSR64_67	0x0080	INTC_PSR176_179	0x00F0
INTC_PSR68_71	0x0084	INTC_PSR180_183	0x00F4

Table 74. INTC Priority Select Register address offsets (continued)

INTC_PSR _{x_x}	Offset Address	INTC_PSR _{x_x}	Offset Address
INTC_PSR72_75	0x0088	INTC_PSR184_187	0x00F8
INTC_PSR76_79	0x008C	INTC_PSR188_191	0x00FC
INTC_PSR80_83	0x0090	INTC_PSR192_195	0x0100
INTC_PSR84_87	0x0094	INTC_PSR196_199	0x0104
INTC_PSR88_91	0x0098	INTC_PSR200_203	0x0108
INTC_PSR92_95	0x009C	INTC_PSR204_207	0x010C
INTC_PSR96_99	0x00A0	INTC_PSR208_211	0x0110
INTC_PSR100_103	0x00A4	INTC_PSR212_215	0x0114
INTC_PSR104_107	0x00A8	INTC_PSR216_219	0x0118
INTC_PSR108_111	0x00AC	INTC_PSR220_221	0x011C

10.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor.

NOTE

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software settable interrupt request, whose PRI_n value in INTC_PSR0–INTC_PSR221 is higher than the PRI value in INTC_CPR, negates before the interrupt request to the processor for that peripheral or software settable interrupt request is acknowledged, the interrupt request to the processor still can assert or will remain asserted for that peripheral or software settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software settable interrupt request. Also, the PRI value in the INTC_CPR will be updated with the corresponding PRI_n value in INTC_PSR n . Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral or, alternatively, setting its mask bit has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

Table 75. Interrupt vectors

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
Core Interrupts					
—	0x0000	16	—	Critical Input (INTC software vector mode) / NMI	Core
—	0x0010	16	—	Machine check / NMI	Core

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
—	0x0020	16	—	Data Storage	Core
—	0x0030	16	—	Instruction Storage	Core
—	0x0040	16	—	External Input (INTC software vector mode)	Core
—	0x0050	16	—	Alignment	Core
—	0x0060	16	—	Program	Core
—	0x0070	16	—	Reserved	Core
—	0x0080	16	—	System call	Core
—	0x0090	96	—	Unused	Core
—	0x00F0	16	—	Debug	Core
—	0x0100	1792	—	Unused	Core
On-Platform Peripheral Interrupts					
0	0x0800	4	—	Software setable flag 0	Software
1	0x0804	4	—	Software setable flag 1	Software
2	0x0808	4	—	Software setable flag 2	Software
3	0x080C	4	—	Software setable flag 3	Software
4	0x0810	4	—	Software setable flag 4	Software
5	0x0814	4	—	Software setable flag 5	Software
6	0x0818	4	—	Software setable flag 6	Software
7	0x081C	4	—	Software setable flag 7	Software
8	0x0820	4	Reserved		
9	0x0824	4	—	Platform Flash Bank 0 Abort Platform Flash Bank 0 Stall Platform Flash Bank 1 Abort Platform Flash Bank 1 Stall Platform Flash Bank 2 Abort Platform Flash Bank 2 Stall Platform Flash Bank 3 Abort Platform Flash Bank 3 Stall	MCM
10	0x0828	4	—	Combined Error	DMA2x
11	0x082C	4	—	Channel 0	DMA2x
12	0x0830	4	—	Channel 1	DMA2x
13	0x0834	4	—	Channel 2	DMA2x
14	0x0838	4	—	Channel 3	DMA2x
15	0x083C	4	—	Channel 4	DMA2x

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
16	0x0840	4	—	Channel 5	DMA2x
17	0x0844	4	—	Channel 6	DMA2x
18	0x0848	4	—	Channel 7	DMA2x
19	0x084C	4	—	Channel 8	DMA2x
20	0x0850	4	—	Channel 9	DMA2x
21	0x0854	4	—	Channel 10	DMA2x
22	0x0858	4	—	Channel 11	DMA2x
23	0x085C	4	—	Channel 12	DMA2x
24	0x0860	4	—	Channel 13	DMA2x
25	0x0864	4	—	Channel 14	DMA2x
26	0x0868	4	—	Channel 15	DMA2x
27	0x086C	4	Reserved		
28	0x0870	4	—	Timeout	Software Watchdog (SWT)
29	0x0874	4	Reserved		
30	0x0878	4	—	Match on channel 0	STM
31	0x087C	4	—	Match on channel 1	STM
32	0x0880	4	—	Match on channel 2	STM
33	0x0884	4	—	Match on channel 3	STM
34	0x0888	4	Reserved		
35	0x088C	4	—	ECC_DBD_PlatformFlash ECC_DBD_PlatformRAM	MCM
36	0x0890	4	—	ECC_SBC_PlatformFlash ECC_SBC_PlatformRAM	MCM
37	0x0894	4	Reserved		
Common module interrupts					
38	0x0898	4	Reserved		
39	0x089C	4	Reserved		
40	0x08A0	4	Reserved		
41	0x08A4	4	GROUP_0	SIU External IRQ_0	System Integration Unit Lite (SIUL)
42	0x08A8	4	GROUP_1	SIU External IRQ_1	System Integration Unit Lite (SIUL)
43	0x08AC	4	GROUP_2	SIU External IRQ_2	System Integration Unit Lite (SIUL)

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
44	0x08B0	4	GROUP_3	SIU External IRQ_3	System Integration Unit Lite (SIUL)
45	0x08B4	4	Reserved		
46	0x08B8	4	Reserved		
47	0x08BC	4	Reserved		
48	0x08C0	4	Reserved		
49	0x08C4	4	Reserved		
50	0x08C8	4	Reserved		
51	0x08CC	4	ME_SAFE_MODE	Safe Mode Interrupt	MC_ME
52	0x08D0	4	ME_MODE_TRANS	Mode Transition Interrupt	MC_ME
53	0x08D4	4	ME_INVALID_MODE	Invalid Mode Interrupt	MC_ME
54	0x08D8	4	ME_INVALID_CONFIG	Invalid Mode Config	MC_ME
55	0x08DC	4	Reserved		
56	0x08E0	4	IRQ	Functional and destructive reset alternate event interrupt (ipi_int)	MC_RGM
57	0x08E4	4	IRQ	XOSC counter expired (ipi_int_osc)	XOSC
58	0x08E8	4	Reserved		
59	0x08EC	4	PIT_0	PITimer Channel 0	Periodic Interrupt Timer (PIT)
60	0x08F0	4	PIT_1	PITimer Channel 1	Periodic Interrupt Timer (PIT)
61	0x08F4	4	PIT_2	PITimer Channel 2	Periodic Interrupt Timer (PIT)
62	0x08F8	4	all	ADC_EOC	Analog to Digital Converter 0 (ADC0)
63	0x08FC	4	wdg_high	ADC_ER	Analog to Digital Converter 0 (ADC0)
64	0x0900	4	wdg_low	ADC_WD	Analog to Digital Converter 0 (ADC0)
65	0x0904	4	CAN_ERROR	FLEXCAN_ESR[ERR_INT]	FlexCan 0 (CAN0)
66	0x0908	4	CAN_WARN	FLEXCAN_ESR_BOFF FLEXCAN_Transmit_Warning FLEXCAN_Receive_Warning	FlexCan 0 (CAN0)
67	0x090C	4	CAN_WAK	FLEXCAN_ESR_WAK	FlexCan 0 (CAN0)
68	0x0910	4	CAN_03_00	FLEXCAN_BUF_00_03	FlexCan 0 (CAN0)
69	0x0914	4	CAN_07_04	FLEXCAN_BUF_04_07	FlexCan 0 (CAN0)
70	0x0918	4	CAN_11_08	FLEXCAN_BUF_08_11	FlexCan 0 (CAN0)
71	0x091C	4	CAN_15_12	FLEXCAN_BUF_12_15	FlexCan 0 (CAN0)

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
72	0x0920	4	CAN_31_16	FLEXCAN_BUF_16_31	FlexCan 0 (CAN0)
73	0x0924	4	Reserved		
74	0x0928	4	DSPI_TFUF_OF	DSPI_SR[TFUF] DSPI_SR[RFOF] DSPI_SR[SPEF]	DSPI 0
75	0x092C	4	DSPI_EOQF	DSPI_SR[EOQF]	DSPI 0
76	0x0930	4	DSPI_TFFF	DSPI_SR[TFFF]	DSPI 0
77	0x0934	4	DSPI_TCF	DSPI_SR[TCF]	DSPI 0
78	0x0938	4	DSPI_RFDF	DSPI_SR[RFDF]	DSPI 0
79	0x093C	4	LINFLEX_INT_RX	LINFlex_RXI	LIN FLEX 0
80	0x0940	4	LINFLEX_INT_TX	LINFlex_TXI	LIN FLEX 0
81	0x0944	4	LINFLEX_ERR	LINFlex_ERR	LIN FLEX 0
82	0x0948	4	Reserved		
83	0x094C	4	Reserved		
84	0x0950	4	Reserved		
85	0x0954	4	Reserved		
86	0x0958	4	Reserved		
87	0x095C	4	Reserved		
88	0x0960	4	Reserved		
89	0x0964	4	Reserved		
90	0x0968	4	Reserved		
91	0x096C	4	Reserved		
92	0x0970	4	Reserved		
93	0x0974	4	Reserved		
94	0x0978	4	DSPI_TFUF_OF	DSPI_SR[TFUF] DSPI_SR[RFOF] DSPI_SR[SPEF]	DSPI 1
95	0x097C	4	DSPI_EOQF	DSPI_SR[EOQF]	DSPI 1
96	0x0980	4	DSPI_TFFF	DSPI_SR[TFFF]	DSPI 1
97	0x0984	4	DSPI_TCF	DSPI_SR[TCF]	DSPI 1
98	0x0988	4	DSPI_RFDF	DSPI_SR[RFDF]	DSPI 1
99	0x098C	4	LINFLEX_INT_RX	LINFlex_RXI	LIN FLEX 1
100	0x0990	4	LINFLEX_INT_TX	LINFlex_TXI	LIN FLEX 1
101	0x0994	4	LINFLEX_ERR	LINFlex_ERR	LIN FLEX 1
102	0x0998	4	Reserved		

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
103-	0x099C	4	Reserved		
104	0x09A0	4	Reserved		
105	0x09A4	4	Reserved		
106	0x09A8	4	Reserved		
107	0x09AC	4	Reserved		
108	0x09B0	4	Reserved		
109	0x09B4	4	Reserved		
110	0x09B8	4	Reserved		
111	0x09BC	4	Reserved		
112	0x09C0	4	Reserved		
113	0x09C4	4	Reserved		
114	0x09C8	4	DSPI_TFUF_OF	DSPI_SR[TFUF] DSPI_SR[RFOF] DSPI_SR[SPEF]	DSPI 2
115	0x09CC	4	DSPI_EOQF	DSPI_SR[EOQF]	DSPI 2
116	0x09D0	4	DSPI_TFFF	DSPI_SR[TFFF]	DSPI 2
117	0x09D4	4	DSPI_TCF	DSPI_SR[TCF]	DSPI 2
118	0x09D8	4	DSPI_RFDF	DSPI_SR[RFDF]	DSPI 2
119	0x09DC	4	Reserved		
120	0x09E0	4	Reserved		
121	0x09E4	4	Reserved		
122	0x09E8	4	Reserved		
123	0x09EC	4	Reserved		
124	0x09F0	4	Reserved		
125	0x09F4	4	—	IBIF	Inter-IC Bus Interface Controller 0 (I2C0)
126	0x09F8	4	—	IBIF	Inter-IC bus interface controller 1 (I2C1)
127	0x09FC	4	PIT_3	PITimer Channel 3	Periodic Interrupt Timer (PIT)
128	0x0A00	4	Reserved		
129	0x0A04	4	Reserved		
130	0x0A08	4	Reserved		
131	0x0A0C	4	Reserved		
132	0x0A10	4	Reserved		

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
133	0x0A14	4		Reserved	
134	0x0A18	4		Reserved	
135	0x0A1C	4		Reserved	
136	0x0A20	4		Reserved	
137	0x0A24	4		Reserved	
138	0x0A28	4		Reserved	
139	0x0A2C	4		Reserved	
140	0x0A30	4		Reserved	
141	0x0A34	4		Reserved	
142	0x0A38	4		Reserved	
143	0x0A3C	4		Reserved	
144	0x0A40	4		Reserved	
145	0x0A44	4		Reserved	
146	0x0A48	4		Reserved	
147	0x0A4C	4		Reserved	
148	0x0A50	4		Reserved	
149	0x0A54	4		Reserved	
150	0x0A58	4		Reserved	
151	0x0A5C	4		Reserved	
152	0x0A60	4		Reserved	
153	0x0A64	4		Reserved	
154	0x0A68	4		Reserved	
155	0x0A6C	4		Reserved	
156	0x0A70	4		Reserved	
MPC5604E-specific interrupts					
157	0x0A74	4	tmr0	TC0IR	eTimer_0
158	0x0A78	4	tmr1	TC1IR	eTimer_0
159	0x0A7C	4	tmr2	TC2IR	eTimer_0
160	0x0A80	4	tmr3	TC3IR	eTimer_0
161	0x0A84	4	tmr4	TC4IR	eTimer_0
162	0x0A8C	4	tmr5	TC5IR	eTimer_0
163	0x0A90	4		Reserved	

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
164	0x0A94	4	Reserved		
165	0x0A98	4	wdog	WTIF	eTimer_0
166	0x0A9C	4	Reserved		
167	0x0AA0	4	rcf	RCF	eTimer_0
168	0x0AA4	4	Reserved		
169	0x0AA8	4	Reserved		
170	0x0AAC	4	Reserved		
171	0x0AB0	4	Reserved		
172	0x0AB4	4	Reserved		
173	0x0AB8	4	Reserved		
174	0x0ABC	4	Reserved		
175	0x0AC0	4	Reserved		
176	0x0AC4	4	Reserved		
177	0x0AC8	4	Reserved		
178	0x0ACC	4	Reserved		
179	0x0AD0	4	encoder	VIS (Vertical Image Start)	VidEnc_0
180	0x0AD4	4	encoder	VIE (Vertical Image End)	VidEnc_0
181	0x0AD8	4	input buffer	SCR (Sub-Channel Ready)	VidEnc_0
182	0x0ADC	4	output buffer	PDR (Package Data Ready)	VidEnc_0
183	0x0AE0	4	all	INT ¹	VidEnc_0
184	0x0AE4	4	all	ERR ²	VidEnc_0
185	0x0AE8	4	all	FEC	FEC_0
186	0x0AEC	4	tx	TX	FEC_0
187	0x0AF0	4	rx	RX	FEC_0
188	0x0AF4	4	cept	TIMESTAMP	PTP
189	0x0AF8	4	ce_rtc	TIMER	CE_RTC
190	0x0AFC	4	tx_fifo	TX	SAI_0
191	0x0B00	4	rx_fifo	RX	SAI_0
192	0x0B04	4	tx_fifo	TX	SAI_1
193	0x0B08	4	rx_fifo	RX	SAI_1
194	0x0B0C	4	tx_fifo	TX	SAI_2
195	0x0B10	4	rx_fifo	RX	SAI_2

Table 75. Interrupt vectors (continued)

IRQ#	Offset	Size [Bytes]	Resource	Interrupt	Module
196	0x0B14	4		Reserved	
197	0x0B18	4		Reserved	
198	0x0B1C	4		Reserved	
199	0x0B20	4		Reserved	
200	0x0B24	4		Reserved	
201	0x0B28	4		Reserved	
202	0x0B30	4		Reserved	
203	0x0B34	4		Reserved	
204	0x0B38	4		Reserved	
205	0x0B3C	4		Reserved	
206	0x0B40	4		Reserved	
207	0x0B44	4		Reserved	
208	0x0B48	4		Reserved	
209	0x0B4C	4		Reserved	
210	0x0B50	4		Reserved	
211	0x0B54	4		Reserved	
212	0x0B58	4		Reserved	
213	0x0B5C	4		Reserved	
214	0x0B60	4		Reserved	
215	0x0B64	4		Reserved	
216	0x0B68	4		Reserved	
217	0x0B6C	4		Reserved	
218	0x0B70	4		Reserved	
219	0x0B74	4		Reserved	
220	0x0B78	4		Reserved	
221	0x0B7C	4		Reserved	

¹ 183 is INT which is a single interrupt line ORing all the interrupts that are generated by the video encoder. This is an Active Low Line.

² 184 is the interrupt for the errors i.e Length error and count error line.
 Len_err_irq(Interrupt signalling mismatch between active line length and programmed line length).
 count err irq(Interrupt signalling mismatch between active image height and programmed image height)

10.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software configurable. These interrupt requests can assert on any clock cycle.

10.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit that resides in the peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC to the time that the INTC starts to drive the interrupt request to the processor is three clocks.

External interrupts are handled by the SIU (see [Section 13.6.3, “External interrupts”](#)).

10.6.1.2 Software configurable interrupt requests

An interrupt request is triggered by software by writing a '1' to a SETx bit in INTC_SSCIR0_3–INTC_SSCIR4_7. This write sets the corresponding flag bit, CLR_x, resulting in the interrupt request. The interrupt request is cleared by writing a '1' to the CLR_x bit.

The time from the write to the SETx bit to the time that the INTC starts to drive the interrupt request to the processor is four clocks.

10.6.1.3 Unique vector for each interrupt request source

Each peripheral and software configurable interrupt request is assigned a hardwired unique 9-bit vector. Software configurable interrupts 0–7 are assigned vectors 0–7 respectively. The peripheral interrupt requests are assigned vectors 8 to as high as needed to include all the peripheral interrupt requests. The peripheral interrupt request input ports at the boundary of the INTC block are assigned specific hardwired vectors within the INTC (see [Table 66](#)).

10.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI_x values set in INTC_PSR0_3–INTC_PSR292_293. The result is compared to PRI in the associated INTC_CPR. The results of those comparisons manage the priority of the ISR executed by the associated processor. The associated LIFO also assists in managing that priority.

10.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator subblocks shown in [Figure 73](#) compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software configurable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software settable interrupt request is generated for INTC interrupt acknowledge register (INTC_IACKR), and if in hardware vector mode, for the interrupt vector provided to the processor.

10.6.2.1.1 Priority arbitrator subblock

The priority arbitrator subblock for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software configurable. The output of the priority arbitrator subblock is the highest of those priorities assigned to a given processor. Also, any interrupt requests that have this highest priority are output as asserted interrupt requests to the associated request selector subblock.

10.6.2.1.2 Request selector subblock

If only one interrupt request from the associated priority arbitrator subblock is asserted, then it is passed as asserted to the associated vector encoder subblock. If multiple interrupt requests from the associated priority arbitrator subblock are asserted, only the one with the lowest vector passes as asserted to the associated vector encoder subblock. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software configurable interrupt requests.

10.6.2.1.3 Vector encoder subblock

The vector encoder subblock generates the unique 9-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

10.6.2.1.4 Priority comparator subblock

The priority comparator submodule compares the highest priority output from the priority arbitrator submodule with PRI in INTC_CPR. If the priority comparator submodule detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in INTC_CPR or the PRI value in INTC_CPR is lowered below this highest priority. This highest priority then becomes the new priority that will be written to PRI in INTC_CPR when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI_n in INTC_PSR_n are zero will not cause a preemption because their PRI_n will not be higher than PRI in INTC_CPR.

10.6.2.2 Last-in first-out (LIFO)

The LIFO stores the preempted PRI values from the INTC_CPR. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the INTC_CPR does not need to be loaded from the INTC_CPR and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the INTC_CPR.

The PRI value in the INTC_CPR is pushed onto the LIFO when the INTC_IACKR is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode. The priority is popped into PRI in the INTC_CPR whenever the INTC_EOIR is written.

Although the INTC supports 16 priorities, an ISR executing with PRI in the INTC_CPR equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 entries deep. An entry for a priority of 0 is not needed because of how pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 15 or more times than it is popped, the priorities

first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop 0s if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

The LIFO is not memory mapped.

10.6.3 Handshaking with processor

10.6.3.1 Software vector mode handshaking

This section describes handshaking in software vector mode.

10.6.3.1.1 Acknowledging interrupt request to processor

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode and the handshake near the end of the interrupt exception handler, is shown in [Figure 82](#). The INTC examines the peripheral and software configurable interrupt requests. When it finds an asserted peripheral or software configurable interrupt request with a higher priority than PRI in the associated INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the associated INTC_IACKR is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of handshaking process is described in [Section 10.4.1.1, "Software vector mode"](#).

10.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC end-of-interrupt register (INTC_EOIR) must be written. When written, the associated LIFO is popped so the preempted priority is restored into PRI of the INTC_CPR. Before it is written, the peripheral or software configurable flag bit must be cleared so that the peripheral or software configurable interrupt request is negated.

NOTE

To ensure proper operation across all eSys MCUs, execute an MBAR or MSYNC instruction between the access to clear the flag bit and the write to the INTC_EOIR.

When returning from the preemption, the INTC does not search for the peripheral or software settable interrupt request whose ISR was preempted. Depending on how much the ISR progressed, that interrupt request may no longer even be asserted. When PRI in INTC_CPR is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it was to next execute before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

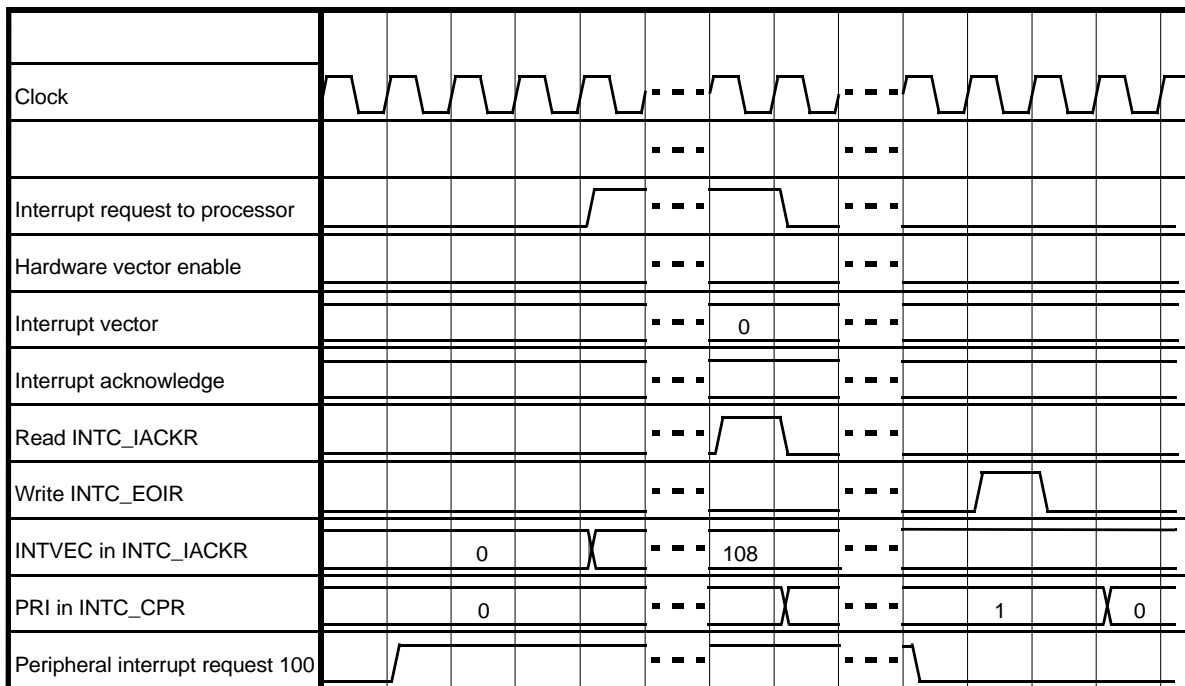


Figure 82. Software vector mode handshaking timing diagram

10.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in [Figure 83](#). As in software vector mode, the INTC examines the peripheral and software settable interrupt requests, and when it finds an asserted one with a higher priority than PRI in INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the INTC_IACKR is updated with the preempting peripheral or software settable interrupt request’s vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. In addition, the value of the interrupt vector to the processor matches the value of the INTVEC field in the INTC_IACKR. The rest of the handshaking is described in” [Section 10.4.1.2, “Hardware vector mode”](#)”.

The handshaking near the end of the interrupt exception handler, that is the writing to the INTC_EOIR, is the same as in software vector mode. Refer to [Section 10.6.3.1.2, “End of interrupt exception handler”](#)”.

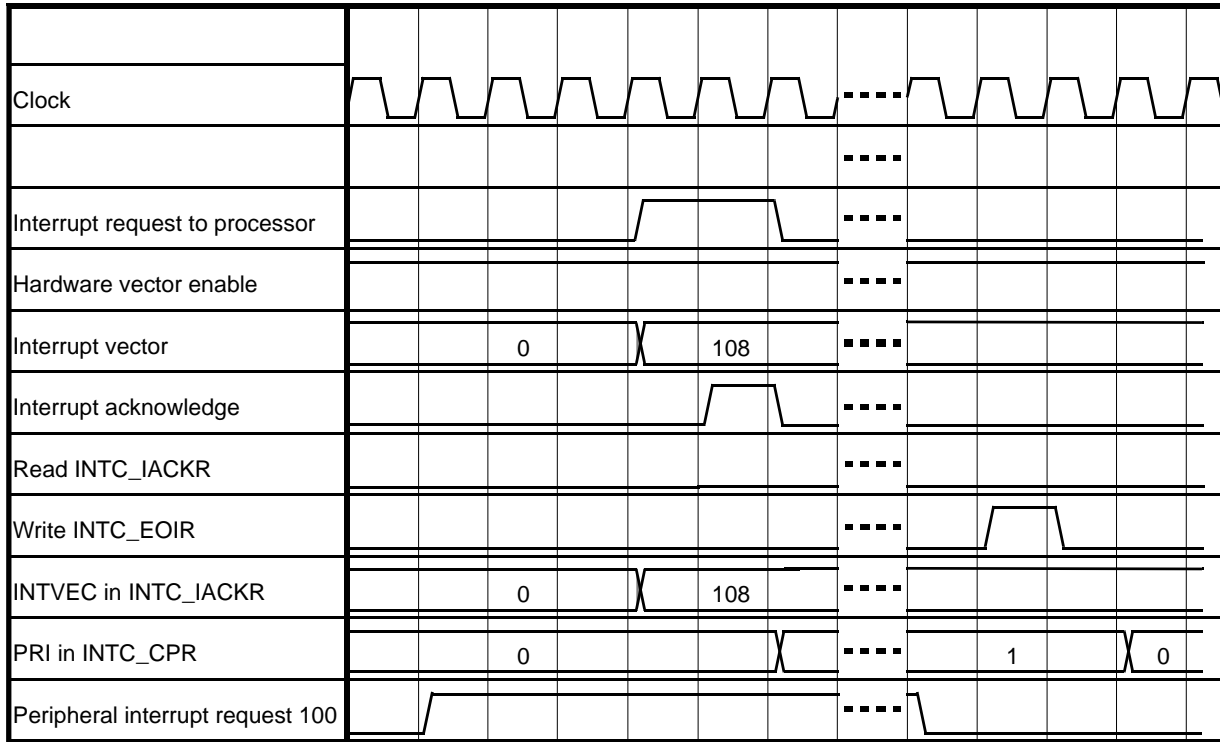


Figure 83. Hardware vector mode handshaking timing diagram

10.7 Initialization/application information

10.7.1 Initialization flow

After exiting reset, all of the PRI_n fields in INTC priority select registers (INTC_PSR0–INTC_PSR211) will be zero, and PRI in INTC current priority register (INTC_CPR) will be 15. These reset values will prevent the INTC from asserting the interrupt request to the processor. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software settable interrupt requests to cause an interrupt request to the processor is: `interrupt_request_initialization`:

```
interrupt_request_initialization:
configure VTES and HVEN in INTC_MCR
configure VTBA in INTC_IACKR
raise the  $PRI_n$  fields in INTC_PSR $n$ 
set the enable bits or clear the mask bits for the peripheral interrupt requests
lower PRI in INTC_CPR to zero
enable processor recognition of interrupts
```

10.7.2 Interrupt exception handler

These example interrupt exception handlers use Power Architecture assembly code.

10.7.2.1 Software vector mode

```

interrupt_exception_handler:
code to create stack frame, save working register, and save SRR0 and SRR1
lis    r3,INTC_IACKR@ha          # form adjusted upper half of INTC_IACKR address
lwz    r3,INTC_IACKR@l(r3)      # load INTC_IACKR, which clears request to processor
lwz    r3,0x0(r3)               # load address of ISR from vector table
wrteei 1                        # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

mtrlr  r3                       # move INTC_IACKR contents into link register
blr    r3                       # branch to ISR; link register updated with epilg
                                           # address

epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                               # ensure store to clear flag bit has completed
lis    r3,INTC_EOIR@ha           # form adjusted upper half of INTC_EOIR address
li     r4,0x0                   # form 0 to write to INTC_EOIR
wrteei 0                          # disable processor recognition of interrupts
stw    r4,INTC_EOIR@l(r3)       # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
.
.
.
address of ISR for interrupt with vector 510
address of ISR for interrupt with vector 511

ISRx:
code to service the interrupt event
code to clear flag bit that drives interrupt request to INTC

blr # return to epilog

```

10.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector. This example assumes that each `interrupt_exception_handlerx` only has space for four instructions, and therefore a branch to `interrupt_exception_handler_continuedx` is needed.

```

interrupt_exception_handlerx:
b interrupt_exception_handler_continuedx# 4 instructions available, branch to continue

```

```

interrupt_exception_handler_continuedx:
code to create stack frame, save working register, and save SRR0 and SRR1

wrteei    1                # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

bl        ISRx            # branch to ISR for interrupt with vector x

epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                      # ensure store to clear flag bit has completed
lis     r3,INTC_EOIR@ha   # form adjusted upper half of INTC_EOIR address
li      r4,0x0           # form 0 to write to INTC_EOIR
wrteei    0                # disable processor recognition of interrupts
stw     r4,INTC_EOIR@l(r3) # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi

ISRx:
code to service the interrupt event
code to clear flag bit that drives interrupt request to INTC
blr                      # branch to epilog
    
```

10.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC current priority register (INTC_CPR) having a value of 0. The RTOS will execute the tasks according to whatever priority scheme that it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR priority 0.

If a task shares a resource with an ISR and the PCP is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR while the shared resource is being accessed.

An ISR whose PRI_n in INTC priority select registers (INTC_PSR0–INTC_PSR211) has a value of 0 will not cause an interrupt request to the processor, even if its peripheral or software settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

10.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software configurable interrupt requests. However, if multiple peripheral or software configurable interrupt requests are asserted, more than one has the highest priority, and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software configurable interrupt requests asserted.

The example in [Table 76](#) shows the order of execution of both ISRs with different priorities and the same priority

Table 76. Order of ISR execution example

Step #	Step Description	Code Executing at End of Step					Interrupt Exception Handler	PRI in INTC_CPR at End of Step
		RTOS	ISR108 ₁	ISR208	ISR308	ISR408		
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 is asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 is asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 is asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR.						X	0
12	RTOS continues execution.	X						0

¹ ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software configurable interrupt requests.

10.7.5 Priority ceiling protocol

10.7.5.1 Elevating priority

The PRI field in INTC_CPR is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR to 3, the ceiling of all of the ISR priorities. After they release the resource, the PRI value in INTC_CPR can be lowered. If they do not raise their priority, ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

10.7.5.2 Ensuring coherency

A scenario can cause non-coherent accesses to the shared resource. For example, ISR1 and ISR2 are both running on the same core and both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing and writes to the INTC_CPR. The instruction following this store is a store to a value in a shared coherent data block. Either immediately before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible that both stores will be executed. ISR2 thereby thinks that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent corruption of a coherent data block, modifications to PRI in INTC_CPR can be made by those system services with the code sequence:

```
disable processor recognition of interrupts
PRI modification
enable processor recognition of interrupts
```

10.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using rate monotonic scheduling (RMS) or a superset of it, deadline monotonic scheduling (DMS). In RMS, the ISRs that have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is

assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 μ s, ISR2 executes every 200 μ s, and ISR3 executes every 300 μ s. ISR1 has a higher priority than ISR2, which has a higher priority than ISR3; however, if ISR3 has a deadline of 150 μ s, then it has a higher priority than ISR2.

The INTC has 16 priorities, which may be less than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 μ s would share a priority, ISRs with request rates around 250 μ s would share a priority, etc. With this approach, a range of ISR request rates of 2^{16} could be included, regardless of the number of ISRs.

Reducing the number of priorities reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They do not need to use the PCP to access the shared resource.

10.7.7 Software configurable interrupt requests

The software configurable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and they may also be used by processors to interrupt other processors in a multiple processor system.

10.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI_x value in INTC_PSR0_3–INTC_PSR292_293, which becomes the PRI value in INTC_CPR with the interrupt acknowledge. The ISR, however, can have a portion that does not need to be executed at this higher priority. Therefore, executing the later portion that does not need to be executed at this higher priority can prevent the execution of ISRs that do not have a higher priority than the earlier portion of the ISR but do have a higher priority than what the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET_x bit in INTC_SSCIR0_3–INTC_SSCIR4_7. Writing a '1' to SET_x causes a software configurable interrupt request. This software configurable interrupt request will usually have a lower PRI_x value in the INTC_PSR $_x_x$ and will not cause preemptive scheduling inefficiencies. After generating a software settable interrupt request, the higher priority ISR completes. The lower priority ISR is scheduled according to its priority. Execution of the higher priority ISR is not resumed after the completion of the lower priority ISR.

10.7.7.2 Scheduling an ISR on another processor

Because the SETx bits in the INTC_SSCIRx_x are memory mapped, processors in multiple-processor systems can schedule ISRs on the other processors. One application is that one processor wants to command another processor to perform a piece of work and the initiating processor does not need to use the results of that work. If the initiating processor is concerned that the processor executing the software configurable ISR has not completed the work before asking it to again execute the ISR, it can check if the corresponding CLRx bit in INTC_SSCIRx_x is asserted before again writing a '1' to the SETx bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. To do this, the first processor writes a '1' to a SETx bit on the second processor. After accessing the block of data, the second processor clears the corresponding CLRx bit and then writes 1 to a SETx bit on the first processor, informing it that it can now access the block of data.

10.7.8 Lowering priority within an ISR

A common method for avoiding preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (see [Section 10.7.7.1, “Scheduling a lower priority portion of an ISR”](#)) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

NOTE

Lowering the PRI value in INTC_CPR within an ISR to below the ISR's corresponding PRI value in INTC_PSR0_3–INTC_PSR292_293 allows more preemptions than the LIFO depth can support.

Therefore, the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

10.7.9 Negating an interrupt request outside of its ISR

10.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits that can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits and their corresponding interrupt requests. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

10.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own. One reason that an ISR clears multiple flag bits is because it serviced those flag bits, and therefore the ISRs for these flag bits do not need to be executed.

10.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside its own ISR due to the side effect of an ISR execution or the intentional clearing a flag bit, the priorities of the peripheral or software configurable interrupt requests for these other flag bits must be selected properly. Their PRI_x values in $INTC_PSR0_3$ – $INTC_PSR292_293$ must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to $INTC_SSCIR0_3$ – $INTC_SSCIR4_7$ as the clearing of the flag bit that caused the present ISR to be executed (see [Section 10.6.3.1.2, “End of interrupt exception handler”](#)).

A flag bit whose enable bit or mask bit negates its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request’s PRI_x value in $INTC_PSR_x_x$.

10.7.10 Examining LIFO contents

In normal mode, the user does not need to know the contents of the LIFO. He may not even know how deeply the LIFO is nested. However, if he wants to read the contents, such as in debug mode, they are not memory mapped. The contents can be read by popping the LIFO and reading the PRI field in either $INTC_CPR$. The code sequence is:

```
pop_lifo:
store to INTC_EOIR
load INTC_CPR, examine PRI, and store onto stack
if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
load stacked PRI value and store to INTC_CPR
load INTC_IACKR
if stacked PRI values are not depleted, branch to push_lifo
```


Chapter 11

Wakeup Unit (WKPU)

11.1 Introduction

11.1.1 Overview

The WKPU supports one external source that can cause non-maskable interrupt requests or wakeup events.

Figure 84 is a block diagram of the WKPU and its interfaces to other system components.

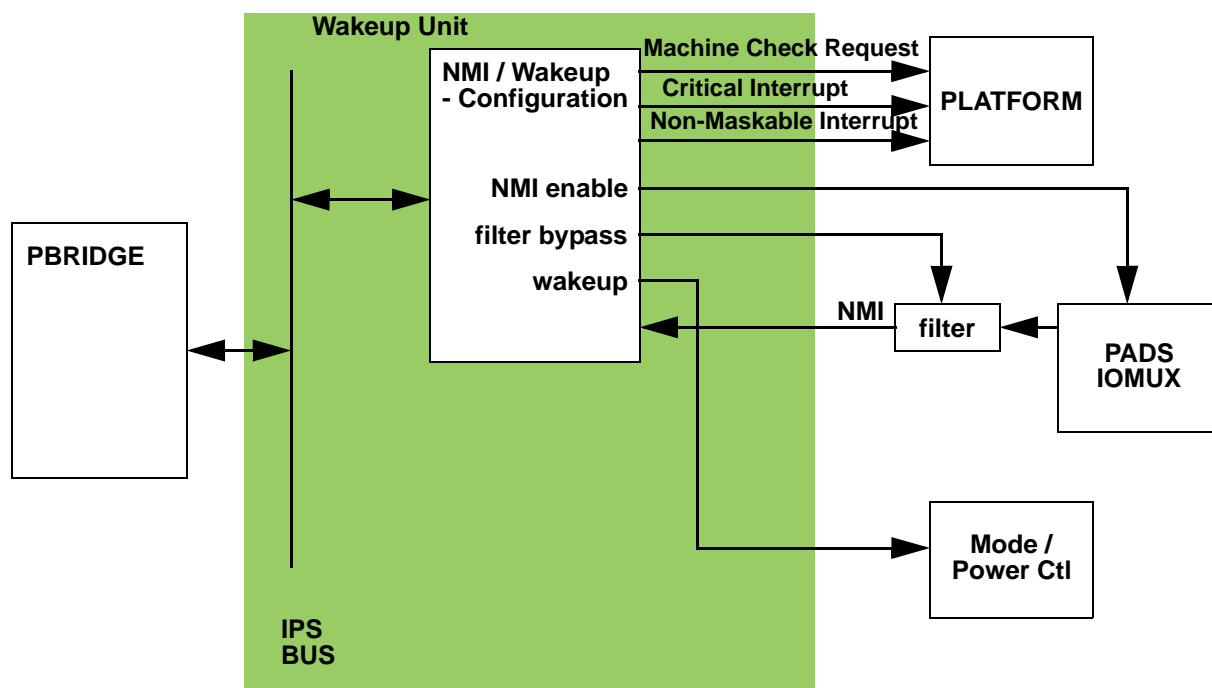


Figure 84. WKPU block diagram

11.1.2 Features

The WKPU supports:

- 1 NMI source
- 1 analog glitch filter
- Independent interrupt destination: non-maskable interrupt, critical interrupt, or machine check request

Wakeup Unit (WKPU)

- Edge detection
- Configurable system wakeup triggering from NMI sources

11.2 External signal description

The module has 1 signal input that can be used as a non-maskable interrupt source in normal run mode or as system wakeup sources in certain power down modes.

11.3 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

11.3.1 Memory map

Figure 77 gives an overview on the WKPU registers implemented.

Table 77. WKPU memory map

Address Offset	Use	Abbreviation	Size	Supported Access Sizes
0x0000	NMI Status Flag Register	NSR	32	32/16/8
0x0004 - 0x0007	Reserved			
0x0008	NMI Configuration Register	NCR	32	32/16/8
0x000C - 0x3FFF	Reserved			

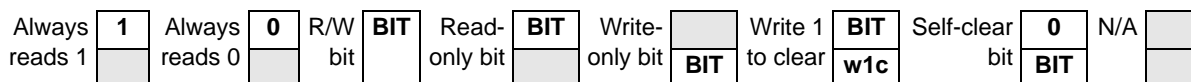
NOTE

Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

11.3.2 Register descriptions

This section describes in address order all the WKPU registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Figure 85. Key to Register Fields



11.3.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Address: 0x0000 Access: User read/write (write 1 to clear)

	0	1	2	3	4	5	6	7
R	NIF0	NOVF0	0	0	0	0	0	0
W	w1c	w1c						
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 86. NMI Status Flag Register (NSR)

Table 78. NSR Field Descriptions

Field	Description
NIF0	NMI Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE0 or NFEE0 set), NIF0 causes an interrupt request. 1 An event as defined by NREE0 and NFEE0 has occurred 0 No event has occurred on the pad
NOVF0	NMI Overrun Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE0 or NFEE0 set), NOVF0 causes an interrupt request. 1 An overrun has occurred on NMI input 0 0 No overrun has occurred on NMI input 0

11.3.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Address: 0x0008

Access: User read/write

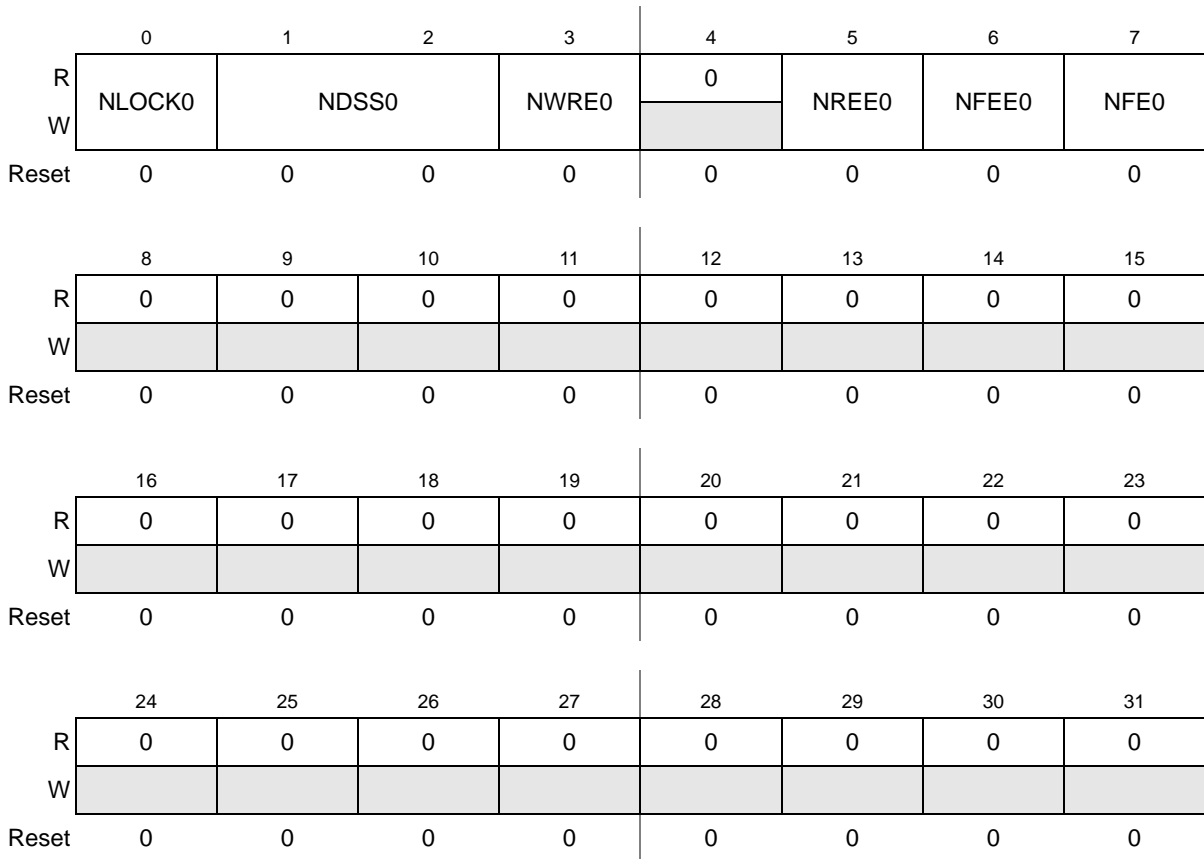


Figure 87. NMI Configuration Register (NCR)

Table 79. NCR Field Descriptions

Field	Description
NLOCK0	NMI Configuration Lock Register 0. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
NDSS0	NMI Destination Source Select 0. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reserved - no NMI, critical interrupt, or machine check request generated
NWRE0	NMI Wakeup Request Enable 0. 1 A set NIF0 bit or set NOV0 bit causes a system wakeup request 0 System wakeup requests from the corresponding NIF0 bit are disabled
NREE0	NMI Rising-edge Events Enable 0. 1 Rising-edge event is enabled 0 Rising-edge event is disabled

Table 79. NCR Field Descriptions (continued)

Field	Description
NFEE0	NMI Falling-edge Events Enable 0. 1 Falling-edge event is enabled 0 Falling-edge event is disabled
NFE0	NMI Filter Enable 0. Enable analog glitch filter on the NMI pad input. 1 Filter is enabled 0 Filter is disabled

NOTE

Writing a '0' to both NREE0 and NFEE0 disables the NMI functionality completely (i.e. no system wakeup or interrupt will be generated on any pad activity)!

11.4 Functional description

11.4.1 General

This section provides a complete functional description of the WKPU.

11.4.2 Non-Maskable Interrupts

The WKPU supports one non-maskable interrupt.

The WKPU supports the generation of 3 types of interrupts per NMI input to the SoC. The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

Each NMI passes through a bypassable analog glitch filter.

NOTE

Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

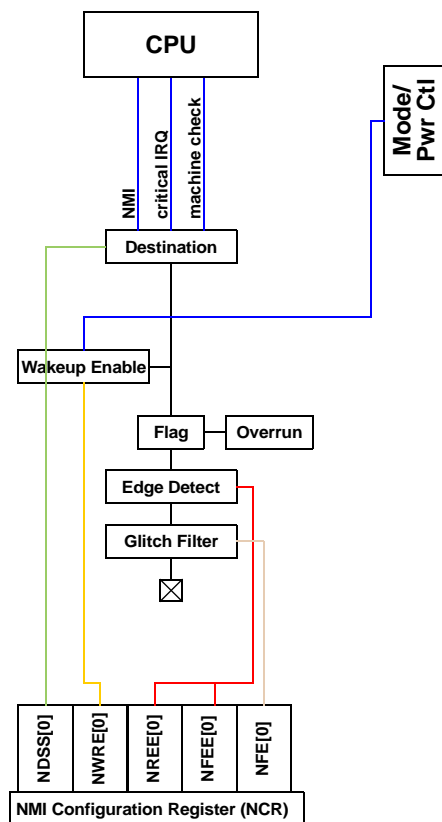


Figure 88. NMI pad diagram

11.4.2.1 NMI management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see Figure 87). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

NOTE

After reset, NREE and NFEE are set to '0', therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See Table 79 for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see Figure 86). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the

same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e. has not yet been cleared).

NOTE

The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 12

System Status and Configuration Module (SSCM)

12.1 Introduction

12.1.1 Overview

The System Status and Configuration Module (SSCM) provides central SOC functionality.

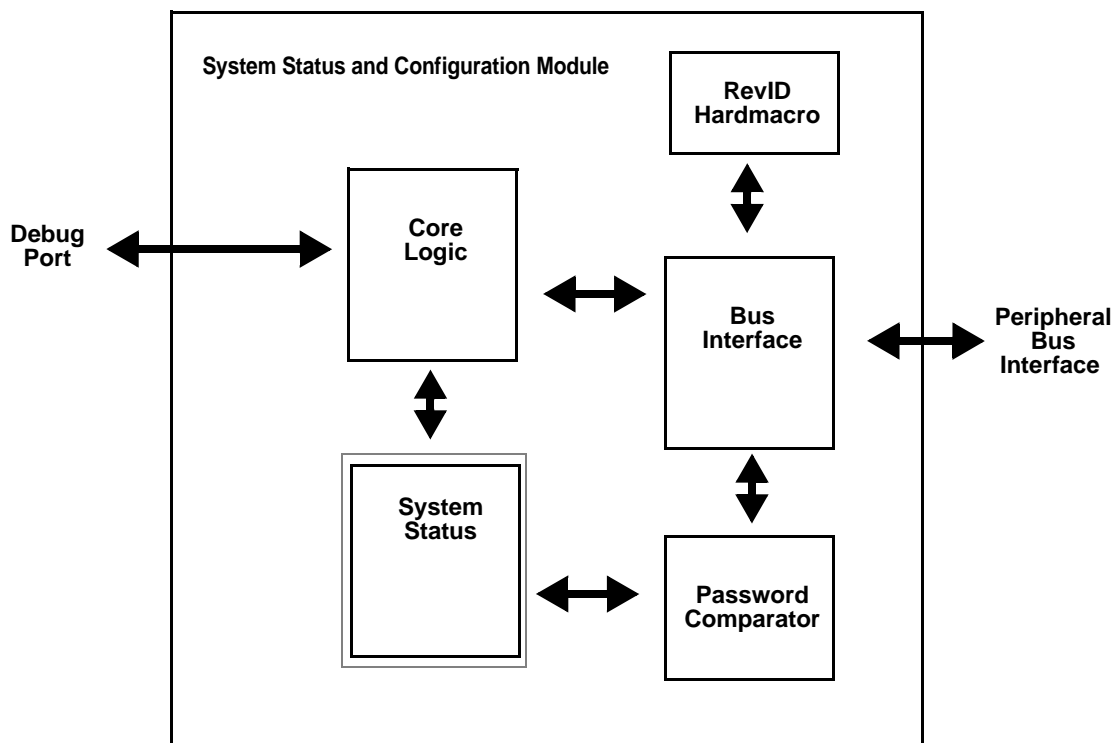


Figure 89. System Status and Configuration Module Block Diagram

12.1.2 Features

The SSCM includes these distinctive features:

- System Configuration and Status
 - Memory sizes/status
 - Device Mode and Security Status
 - Determine boot vector

- Search Code Flash for bootable sector
- DMA Status
- Device identification information (MCU ID Registers)
- Debug Status Port enable and selection
- Bus and peripheral abort enable/disable

12.1.3 Modes of Operation

The SSCM operates identically in all system modes.

12.2 External Signal Description

The SSCM has no external pins.

12.3 Memory Map/Register Definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 80 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Table 80. Module Memory Map

Address	Register	Size	Access	Mode ¹
Base + 0x0000	System Status (STATUS)	16 bits	R	A
Base + 0x0002	System Memory and ID (MEMCONFIG)	16 bits	R	A
Base + 0x0004	<i>Reserved</i>	<i>16 bits</i>	<i>Reads/Writes have no effect.</i>	<i>A</i>
Base + 0x0006	Error Configuration (ERROR)	16 bits	R/W	A
Base + 0x0008	Debug Status Port (DEBUGPORT)	16 bits	R/W	A
Base + 0x000A	<i>Reserved</i>	<i>16 bits</i>	<i>Reads/Writes have no effect.</i>	<i>A</i>
Base + 0x0014 to Base + 0x001C	<i>Reserved</i>	<i>32 bits</i>	<i>Reads/Writes have no effect.</i>	<i>A</i>
Base + 0x0028	Primary Boot Address	32 bits	R	A
Base + 0x002C	<i>Reserved</i>	<i>32 bits</i>	<i>Reads/Writes have no effect.</i>	<i>A</i>

¹ **U** = User Mode, **S** = Supervisor Mode, **T** = Test Mode, **V** = DFV Mode, **A** = All (No restrictions)

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to address 'Base + 0x0002', but performing a 16-bit access to 'Base + 0x0003' is illegal.

12.3.1 Register Descriptions

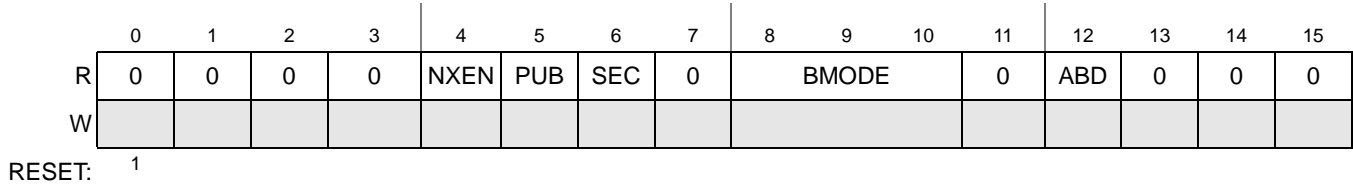
The following registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

12.3.1.1 System Status Register

The System Status register is a read-only register that reflects the current state of the system.

Address: Base + 0x0000

Access: Read / Write



= Reserved

Figure 90. Status (STATUS) Register

¹ Reset values for this register depend on the associated option bits, or on the device status after leaving reset.

Table 81. STATUS Allowed Register Accesses

	8-bit	16-bit	32-bit ¹
READ	Allowed	Allowed	Allowed
WRITE	Allowed	Allowed	Allowed

¹ All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

Table 82. STATUS Field Descriptions

Field	Description
NXEN	Nexus enabled.
PUB	Public Serial Access Status. This bit indicates whether serial boot mode with public password is allowed. 1 Serial boot mode with public password is allowed 0 Serial boot mode with private Flash password is allowed, provided the key hasn't been swallowed
SEC	Security Status. This bit reflects the current security state of the Flash. 1 The Flash is secured 0 The Flash is not secured

Table 82. STATUS Field Descriptions (continued)

Field	Description
BMODE	Device Boot Mode. 000 (reserved for FlexRay Boot Serial Boot Loader) 001 CAN Serial Boot Loader 010 SCI Serial Boot Loader 011 Single Chip 100 Expanded Chip This field is only updated during reset. If the device goes into standby mode and wakes up from it again, the bits retain their original value.
ABD	Autobaud. Indicates that autobaud detection is active when in SCI or CAN serial boot loader mode. No meaning in other modes.

12.3.1.2 System Memory and ID Register

The System Memory Configuration register is a read-only register that reflects the memory configuration of the system. It also contains the JTAG ID.

Address Base + 0x0002

Access: Read Only

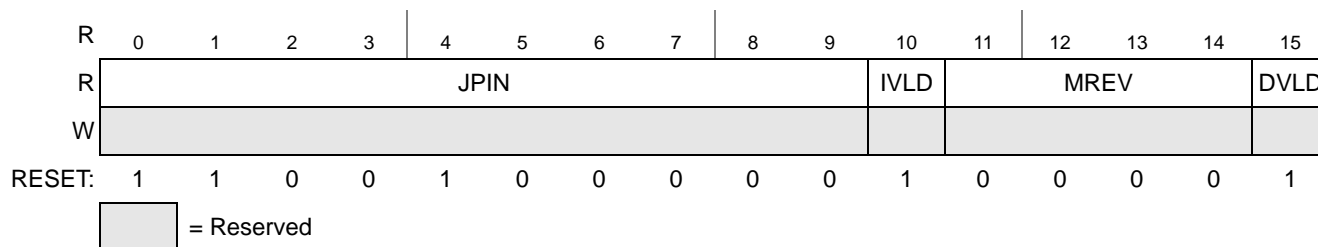


Figure 91. System Memory and ID (MEMCONFIG) Register

Table 83. MEMCONFIG Field Descriptions

Field	Description
JPIN	JTAG Part ID Number
IVLD	Instruction Flash Valid. This bit identifies whether or not the on-chip Instruction Flash is accessible in the system memory map. The Flash may not be accessible due to security limitations, or because there is no Flash in the system. 1 Instruction Flash is accessible 0 Instruction Flash is not accessible
MREV	Minor Mask Revision
DVLD	Data Flash Valid. This bit identifies whether or not the on-chip Data Flash is visible in the system memory map. The Flash may not be accessible due to security limitations, or because there is no Flash in the system. 1 Data Flash is visible 0 Data Flash is not visible

Table 84. MEMCONFIG Allowed Register Accesses

	8-bit	16-bit	32-bit
READ	Allowed	Allowed	Allowed (also reads STATUS register)
WRITE	Not Allowed	Not Allowed	Not Allowed

12.3.1.3 Error Configuration

The Error Configuration register is a read-write register that controls the error handling of the system.

Address : Base + 0x0006

Access: Read/Write

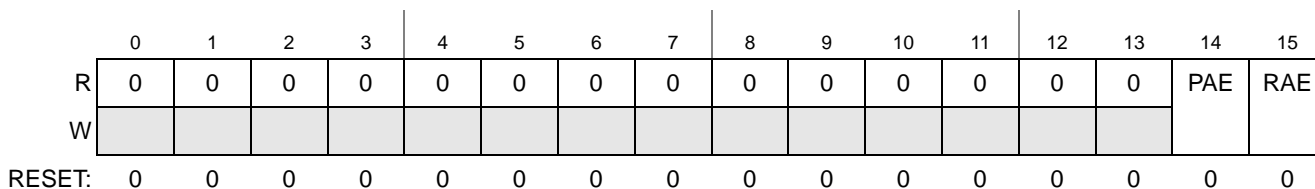


Figure 92. Error Configuration (ERROR) Register

Table 85. ERROR Field Descriptions

Field	Description
PAE	Peripheral Bus Abort Enable. This bit enables bus aborts on any access to a peripheral slot that is not used on the device. This feature is intended to aid in debugging when developing application code. 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception
RAE	Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception Note: Transfers to Peripheral Bus resources may be aborted even before they reach the Peripheral Bus (i.e. at the AIPS level). In this case, the PER_ABORT and REG_ABORT register bits will have no effect on the abort.

Table 86. ERROR Allowed Register Accesses

	8-bit	16-bit	32-bit
READ	Allowed	Allowed	Allowed
WRITE	Allowed	Allowed	Not Allowed

12.3.1.4 Debug Status Port Register

The Debug Status Port register is used to (optionally) provide debug data on a set of pins. Consult the SOC guide for this information.

Address: Base + 0x0008

Access: Read/Write

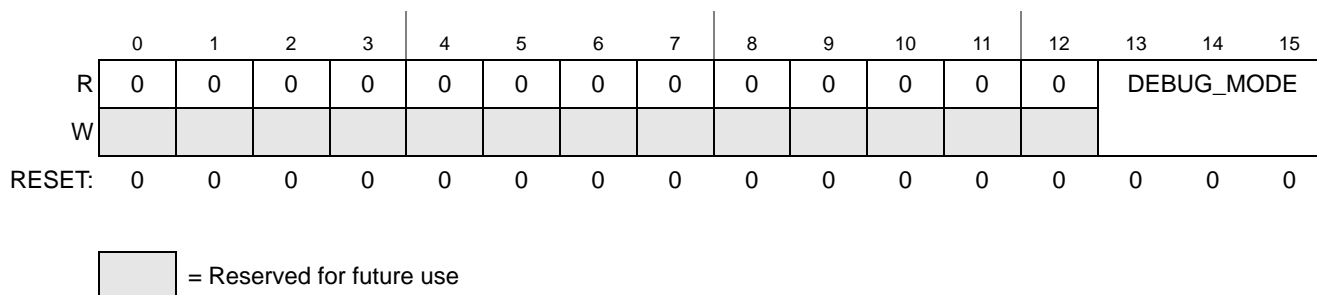


Figure 93. Debug Status Port (DEBUGPORT) Register

Table 87. DEBUGPORT Field Descriptions

Field	Description
DEBUG_MODE	Debug Status Port Mode. This field selects the alternate debug functionality for the Debug Status Port 000 undefined 001 Mode 1 Selected 010 Mode 2 Selected 011 Mode 3 Selected 100 Mode 4 Selected 101 Mode 5 Selected 110 Mode 6 Selected 111 Mode 7 Selected Table 88 describes the functionality of the Debug Status Port in each mode.

Table 88. Debug Status Port Modes

Pin ¹	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
0	STATUS[0]	STATUS[8]	MEMCONFIG[0]	MEMCONFIG[8]	Reserved	Reserved	Reserved
1	STATUS[1]	STATUS[9]	MEMCONFIG[1]	MEMCONFIG[9]	Reserved	Reserved	Reserved
2	STATUS[2]	STATUS[10]	MEMCONFIG[2]	MEMCONFIG[10]	Reserved	Reserved	Reserved
3	STATUS[3]	STATUS[11]	MEMCONFIG[3]	MEMCONFIG[11]	Reserved	Reserved	Reserved
4	STATUS[4]	STATUS[12]	MEMCONFIG[4]	MEMCONFIG[12]	Reserved	Reserved	Reserved
5	STATUS[5]	STATUS[13]	MEMCONFIG[5]	MEMCONFIG[13]	Reserved	Reserved	Reserved
6	STATUS[6]	STATUS[14]	MEMCONFIG[6]	MEMCONFIG[14]	Reserved	Reserved	Reserved
7	STATUS[7]	STATUS[15]	MEMCONFIG[7]	MEMCONFIG[15]	Reserved	Reserved	Reserved

¹ All signals are active high, unless otherwise noted

Table 89. DEBUGPORT Allowed Register Accesses

	8-bit	16-bit	32-bit ¹
READ	Allowed	Allowed	Not Allowed
WRITE	Allowed	Allowed	Not Allowed

¹ All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

Table 90. Debug Status Port Signals

Debug Status Port	Signals
DS6	sscm__ipg_debug_soc__wire
	platform__zcor_pstat__wire[6:0]
DS7	video_wrap__valid_frame__wire
	video_wrap__sub_start__wire
	video_wrap__seq_luma__wire
	video_wrap__seq_chroma__wire dflash0__f90_done__wire
	cflash0__f90_done__wire
	vreg_dig0__vreg_ok__wire
	fmp1l0__i_lock__wire
	fmp1l0__i_lock__wire

12.3.1.5 Primary Boot Address

Address: Base + 0x28

Access: Read/Write

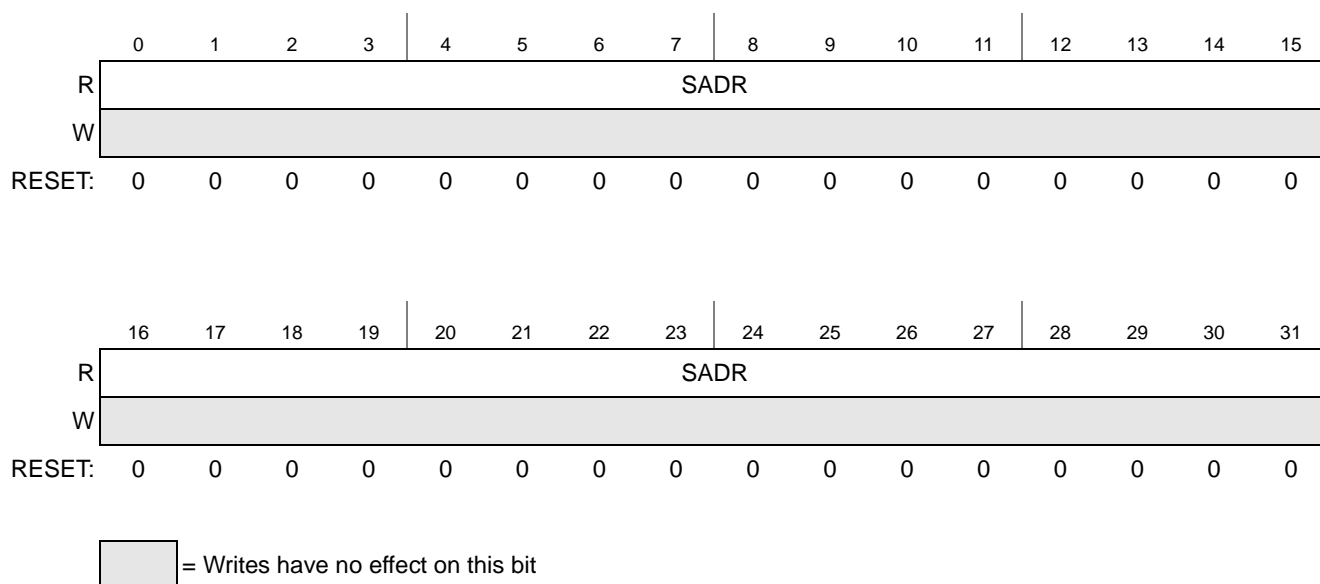


Figure 94.

Table 91. PSA Field Descriptions

Field	Description
SADR	Start Address - the boot processor will start executing application code from this address

12.4 Functional Description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

12.5 Initialization/Application Information

12.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see [Section 12.3.1, “Register Descriptions”](#)).

Chapter 13

System Integration Unit Lite (SIUL)

13.1 Introduction

This chapter describes the System Integration Unit Lite (SIUL), which is used for the management of the pads and their configuration. It controls the multiplexing of the alternate functions used on all pads as well as being responsible for the management of the external interrupts to the device.

13.2 Overview

The System Integration Unit Lite (SIUL) controls the MCU pad configuration, ports, general-purpose input and output (GPIO) signals and external interrupts with trigger event configuration. [Figure 95](#) provides a block diagram of the SIUL and its interfaces to other system components.

The module provides dedicated general-purpose pads that can be configured as either inputs or outputs.

- When a pad is configured as an input, the state of the pad (logic high or low) is obtained by reading an associated data input register.
- When a pad is configured as an output, the value driven onto the pad is determined by writing to an associated data output register. Enabling the input buffers when a pad is configured as an output allows the actual state of the pad to be read.
- To enable monitoring of an output pad value, the pad can be configured as both output and input so the actual pad value can be read back and compared with the expected value.Rev. 5

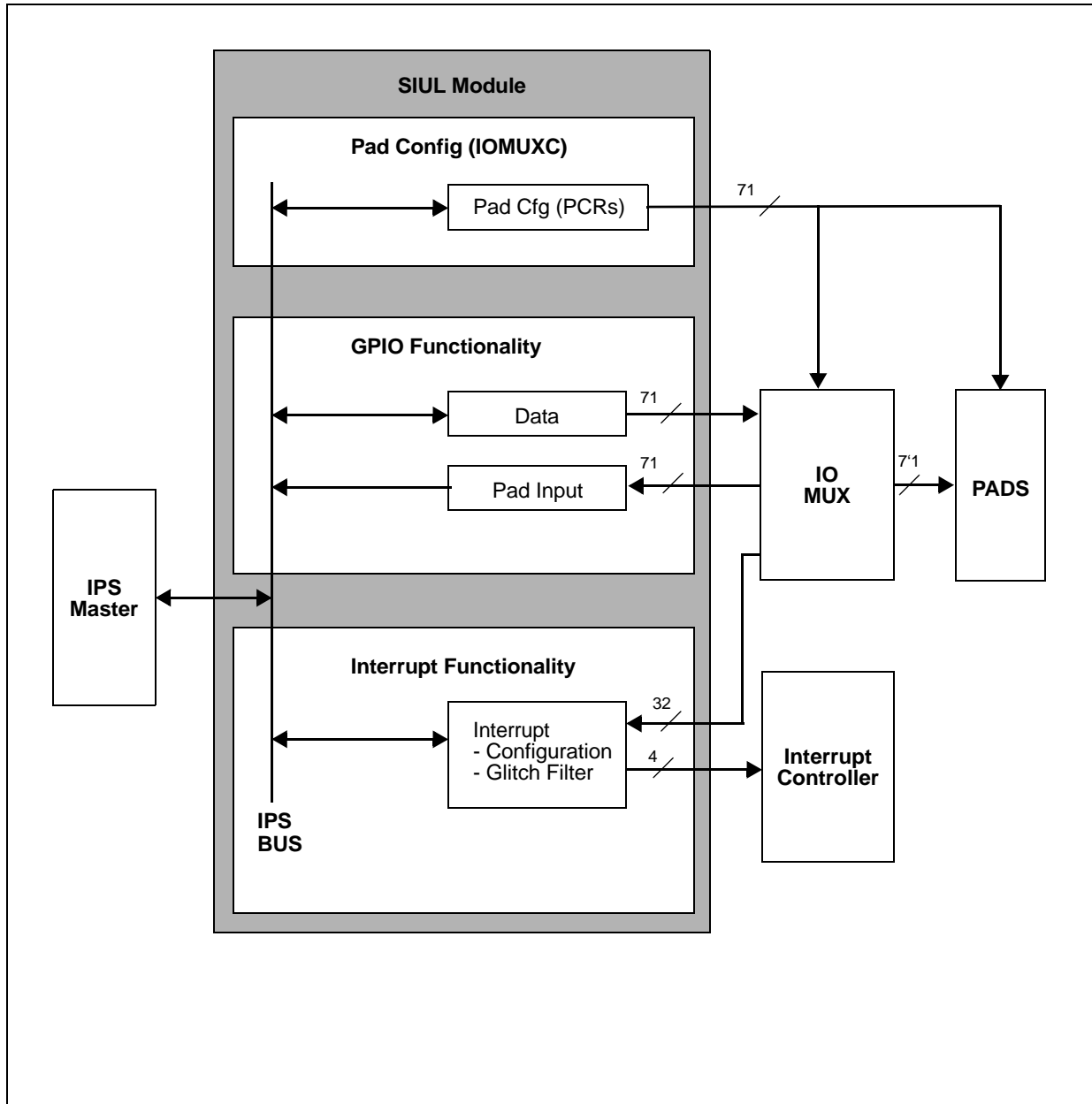


Figure 95. System Integration Unit Lite block diagram

13.3 Features

The System Integration Unit Lite supports these distinctive features:

The System Integration Unit Lite provides these features:

- GPIO
 - GPIO function on up to 71 I/O pins
 - Dedicated input and output registers for each GPIO pin
- External interrupts

- 4 system interrupt vectors for up to 32 interrupt sources
- 32 programmable digital glitch filters
- Independent interrupt mask
- Edge detection
- System configuration
 - Pad configuration control

13.4 External signal description

Most device pads support multiple device functions. Pad configuration registers are provided to enable selection between GPIO and other signals. These other signals, also referred to as alternate functions, are typically peripheral functions.

GPIO pads are grouped in “ports”, with each port containing up to 16 pads. With appropriate configuration, all pins in a port can be read or written to in parallel with a single R/W access.

Table 92 lists the external pins used by the SIUL.

Table 92. SIUL signal properties

GPIO[0:198] category	Name	I/O direction	Function
System configuration	GPIO[0:70]	I/O	General-purpose input/output
External interrupt	A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A12, A13, A14, A15, B3, C1, C2, C3, C4, C5, C6, C8, C9, C10, C12, C13, C14, D9, D13, D14, and E2	Input	Pins with External Interrupt Request functionality. Please refer to the signal description chapter of this reference manual for details.

13.4.1 Detailed signal descriptions

13.4.1.1 General-purpose I/O pins (GPIO[0:70])

The GPIO pins provide general-purpose input and output function. The GPIO pins are generally multiplexed with other I/O pin functions. Each GPIO input and output is separately controlled by an input (GPDIn_n) or output (GPDO_n) register.

13.4.1.2 External interrupt request input pins (EIRQ[0:31])

The EIRQ[0:31] pins are connected to the SIUL inputs. Rising- or falling-edge events are enabled by setting the corresponding bits in the SIUL_IREER or the SIUL_IFEER register.

13.5 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL module.

13.5.1 SIUL memory map

Table 93 gives an overview of the SIUL registers implemented.

Table 93. SIUL memory map

Register Offset	Register name	Location
0xC3F9_0000	Reserved	—
0x0004	MCU ID Register #1 (MIDR1)	on page 225
0x0008	MCU ID Register #2 (MIDR2)	on page 227
0x000C–0x0013	Reserved	—
0x0014	Interrupt Status Flag Register (ISR)	on page 228
0x0018	Interrupt Request Enable Register (IRER)	on page 228
0x001C–0x0027	Reserved	—
0x0028	Interrupt Rising-Edge Event Enable Register (IREER)	on page 229
0x002C	Interrupt Falling-Edge Event Enable Register (IFEER)	on page 229
0x0030	Interrupt Filter Enable Register (IFER)	on page 230
0x0034–0x003F	Reserved	—
0x0040–0x00CC	Pad Configuration Registers (PCR[0:70])	on page 230
0x00CE–0x04FF	Reserved	—
0x500–0x519	Pad Selection for Multiplexed Inputs Registers (PSMI0–PSMI25)	on page 232
0x0520–0x05FF	Reserved	—
0x0600–0x0644	GPIO Pad Data Output Registers (GPDO0_3–GPDO68_71)	on page 235
0x06C8–0x07FF	Reserved	—
0x0800–0x0844	GPIO Pad Data Input Registers (GPDI0_3–GPDI68_71)	on page 236
0x08C8–0x0BFF	Reserved	—
0x0C00–0x0C08	Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO2)	on page 237
0x0C0C–0x0C3F	Reserved	—
0x0C40–0x0C48	Parallel GPIO Pad Data In Register (PGPDI0–PGPDI2)	on page 237
0x0C4C–0x0C7F	Reserved	—
0x0C80–0x0C90	Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO4)	on page 238
0x0C94–0x0FFF	Reserved	—
0x1000–0x107C	Interrupt Filter Maximum Counter Registers (IFMC0–IFMC31)	on page 239
0x1080	Interrupt Filter Clock Prescaler Register (IFCPR)	on page 241
0x1084–0x3FF	Reserved	—

NOTE

A transfer error will be issued when trying to access completely reserved register space.

13.5.2 Register protection

Individual registers in System Integration Unit Lite can be protected from accidental writes using the Register Protection module ([Chapter 39, "Register Protection \(REG_PROT\)"](#)). The following registers can be protected:

- Interrupt Request Enable Register (IRER)
- Interrupt Rising-Edge Event Enable Register (IREER)
- Interrupt Falling-Edge Event Enable Register (IFEER)
- Interrupt Filter Enable Register (IFER)
- Interrupt Filter Enable Register (IFER)
- Pad Configuration Registers (PCR[0:70])
- Pad Selection for Multiplexed Inputs Registers (PSMI0–PSMI25)
- Interrupt Filter Maximum Counter Registers (IFMC0–IFMC31)
- Interrupt Filter Clock Prescaler Register (IFCPR)

Refer to [Chapter 39, "Register Protection \(REG_PROT\)"](#) for details.

13.5.3 Register description

This section describes in address order all the SIUL registers. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order. The numbering convention of the registers is MSB = 0, however the numbering of the internal fields is LSB = 0, for example, PARTNUM[5] = MIDR1[10].

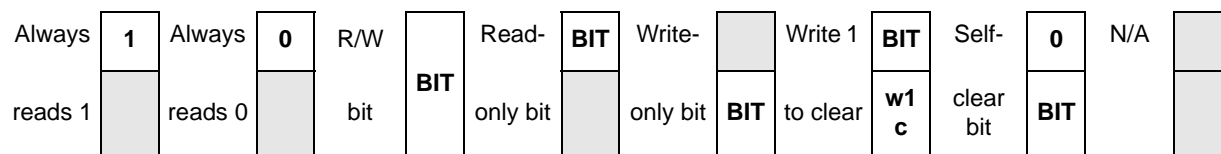


Figure 96. Key to register fields

13.5.3.1 MCU ID Register #1 (MIDR1)

This register contains the part number and the package ID of the device.

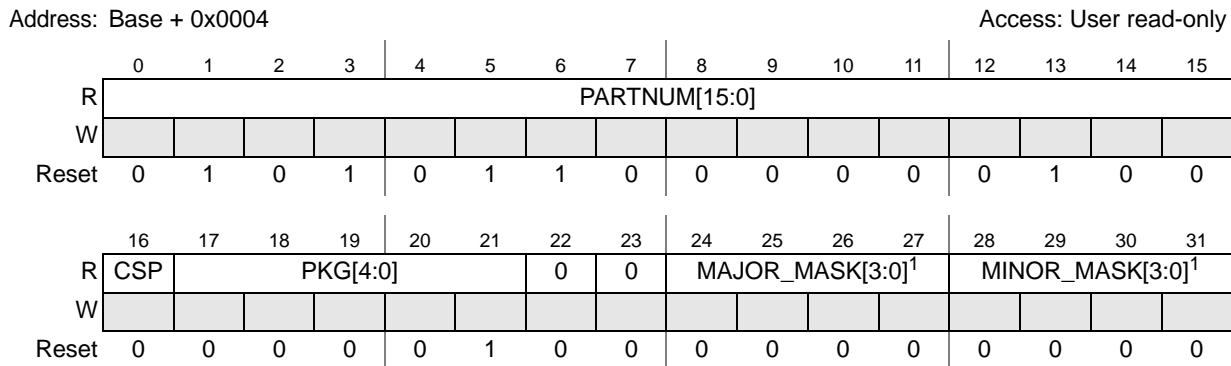


Figure 97. MCU ID Register #1 (MIDR1)

¹ See Table 94.

Table 94. MIDR1 field descriptions

Field	Description
PARTNUM[15:0]	MCU Part Number Device part number of the MCU. 0101_0110_0000_0010: 5602 (ENET) 0101_0110_0000_0011: 5603 (ENET, SAI) 0101_0110_0000_0100: 5604 (ENET, SAI, MPJPEG) For the full part number this field needs to be combined with MIDR2.PARTNUM[23:16]
CSP	Always reads back 0
PKG[4:0]	Package Settings Can be read by software to determine the package type that is used for the particular device: 00001: 64-pin LQFP
MAJOR_MASK[3:0]	Major Mask Revision Counter starting at 0x0. Incremented each time a resynthesis is done. 0b0000
MINOR_MASK[3:0]	Minor Mask Revision Counter starting at 0x0. Incremented each time a mask change is done. 0000: Silicon Cut 1.0 0001: Silicon Cut 1.1 0010: Silicon Cut 1.2

13.5.3.2 MCU ID Register #2 (MIDR2)

This register contains additional configuration information about the device.

Address: Base + 0x0008

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE[3:0]			0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM[23:16]								0	0	0	EE	0	0	0	0
W																
Reset	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0

Figure 98. MCU ID Register #2 (MIDR2)

Table 95. MIDR2 field descriptions

Field	Description
SF	Manufacturer 0: Freescale 1: Reserved
FLASH_SIZE[3:0]	Coarse granularity for Flash memory size 0101: 512 KB Other values are reserved.
PARTNUM[23:16]	ASCII character in MCU Part Number 0x45: ascii 'E' (MPC5604E)
EE	Data Flash present 0: No Data Flash present 1: Data Flash present

13.5.3.3 Interrupt Status Flag Register (ISR)

This register holds the interrupt flags.

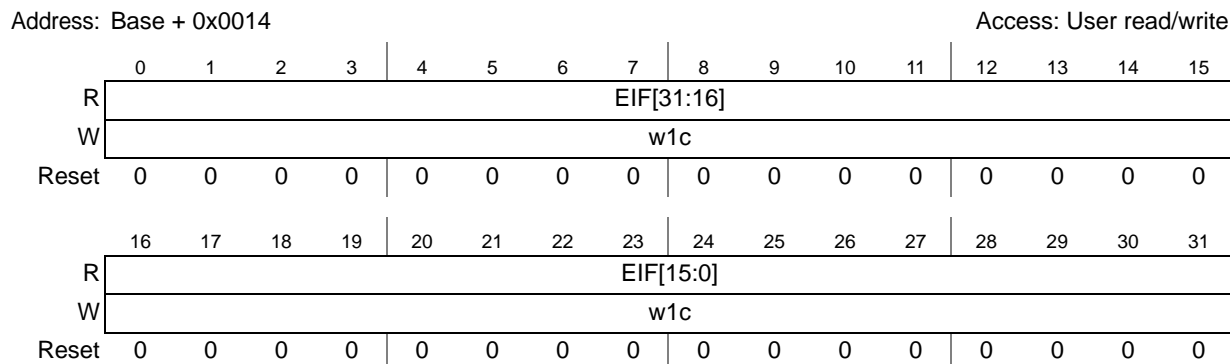


Figure 99. Interrupt Status Flag Register (ISR)

Table 96. ISR field descriptions

Field	Description
EIF n	External Interrupt Status Flag n This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER n), EIF n causes an interrupt request. 0: No interrupt event has occurred on the pad. 1: An interrupt event as defined by IREER n and IFEER n has occurred.

13.5.3.4 Interrupt Request Enable Register (IRER)

This register enables the interrupt messaging to the interrupt controller.

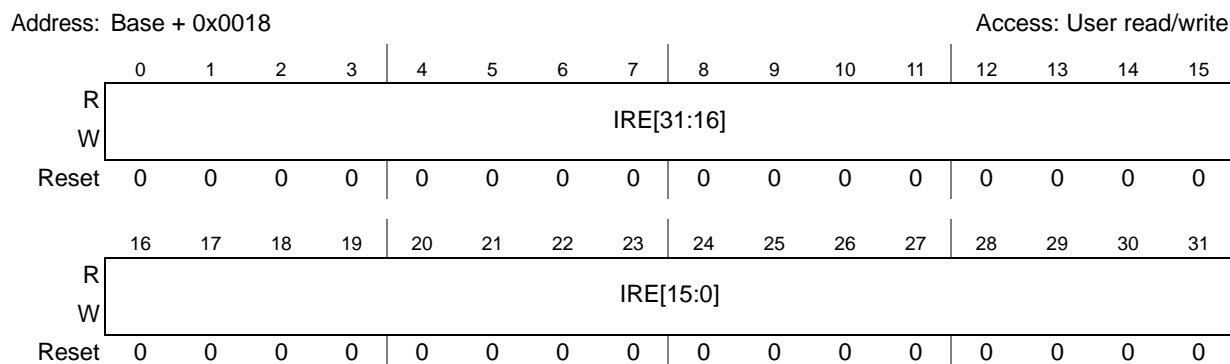


Figure 100. Interrupt Request Enable Register (IRER)

Table 97. IRER field descriptions

Field	Description
IRE n	External Interrupt Request Enable n 0: Interrupt requests from the corresponding EIF n bit are disabled. 1: A set EIF n bit causes an interrupt request.

13.5.3.5 Interrupt Rising-Edge Event Enable Register (IREER)

This register allows rising-edge triggered events to be enabled on the corresponding interrupt pads.

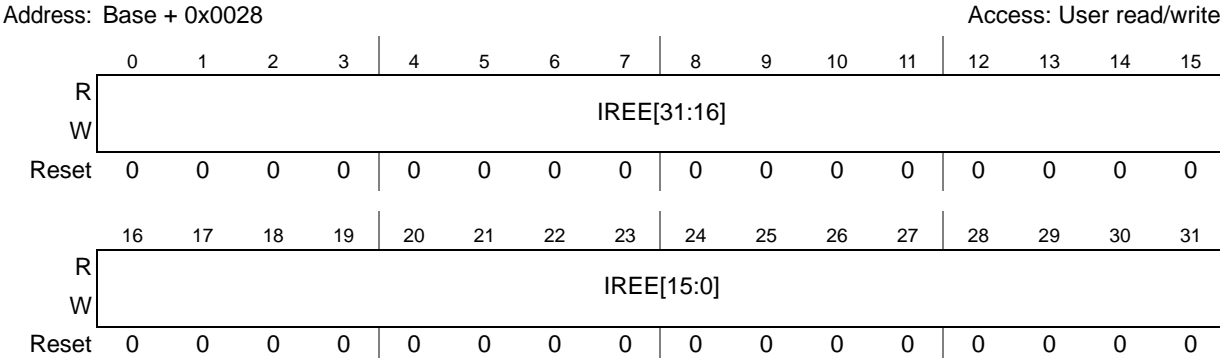


Figure 101. Interrupt Rising-Edge Event Enable Register (IREER)

Table 98. IREER field descriptions

Field	Description
IREE n	Enable rising-edge events to cause the EIF n bit to be set. 0: Rising-edge event disabled 1: Rising-edge event enabled

13.5.3.6 Interrupt Falling-Edge Event Enable Register (IFEER)

This register allows falling-edge triggered events to be enabled on the corresponding interrupt pads.

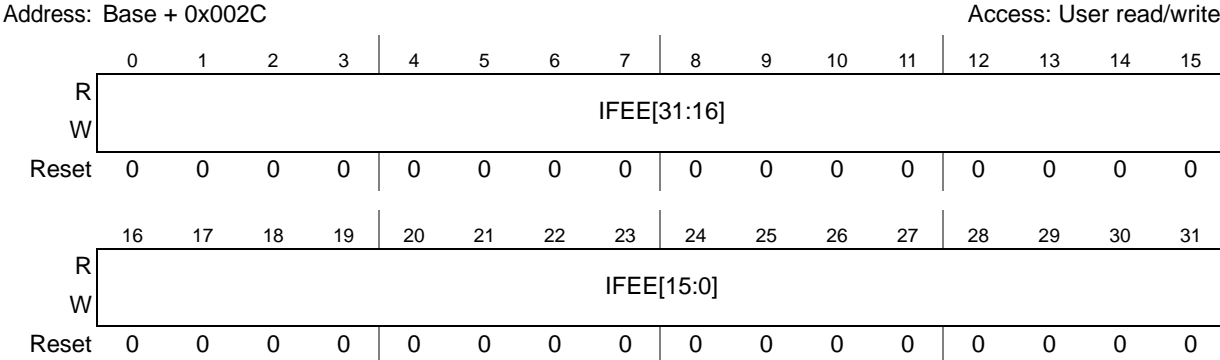


Figure 102. Interrupt Falling-Edge Event Enable Register (IFEER)

Table 99. IFEER field descriptions

Field	Description
IFEE n	Enable falling-edge events to cause the EIF n bit to be set. 0: Falling-edge event disabled 1: Falling-edge event enabled

NOTE

If both the IREER.IREE and IFEER.IFEE bits are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

13.5.3.7 Interrupt Filter Enable Register (IFER)

This register enables a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

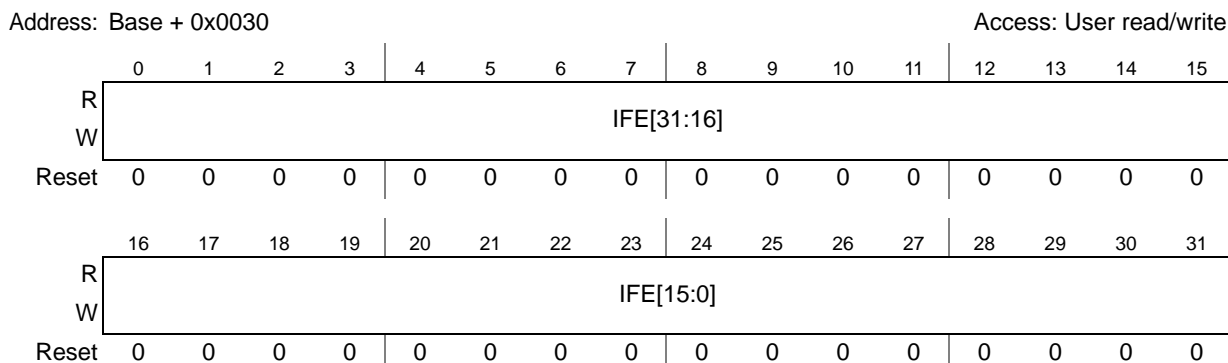


Figure 103. Interrupt Filter Enable Register (IFER)

Table 100. IFER field descriptions

Field	Description
IFERn	Enable digital glitch filter on the interrupt pad input. 0: Filter disabled 1: Filter enabled

13.5.3.8 Pad Configuration Registers (PCR[0:70])

The Pad Configuration Registers allow configuration of the static electrical and functional characteristics associated with I/O pads. Each PCR controls the characteristics of a single pad.

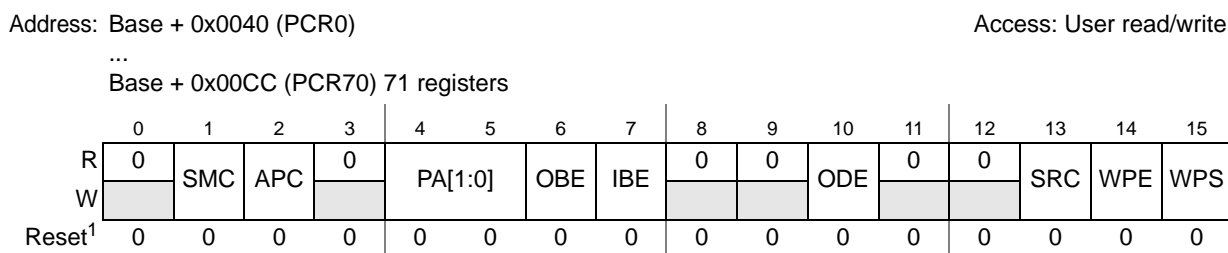


Figure 104. Pad Configuration Registers 0–70 (PCR[0:70])

¹ See Table 102.

NOTE

16/32-bit access is supported for the PCR[0:70] registers.

Table 101. PCR[0:70] field descriptions

Field	Description
SMC	Safe Mode Control This bit supports the overriding of the automatic deactivation of the output buffer of the associated pad upon entering Safe mode of the device. 0: In Safe mode, output buffer of the pad disabled 1: In Safe mode, output buffer remains functional
APC	Analog Pad Control This bit enables the usage of the pad as analog input. 0: Analog input path from the pad is gated and cannot be used. 1: Analog input path switch can be enabled by the ADC.
PA[1:0]	Pad Output Assignment This field selects the function that is allowed to drive the output of a multiplexed pad. The PA field size can vary from 0 to 2 bits, depending on the number of output functions associated with this pad. 00: Alternative mode 0: GPIO 01: Alternative mode 1 (see Signal Description) 10: Alternative mode 2 (see Signal Description) 11: Alternative mode 3 (see Signal Description) Note: The number of bits in the PA bitfield depends of the number of actual alternate functions provided for each pad. Please see the <i>MPC5604E Microcontroller Data Sheet</i>
OBE	Output Buffer Enable This bit enables the output buffer of the pad in case the pad is in GPIO mode. 0: Output buffer of the pad disabled when PA = 00 1: Output buffer of the pad enabled when PA = 00
IBE	Input Buffer Enable This bit enables the input buffer of the pad. 0: Input buffer of the pad disabled 1: Input buffer of the pad enabled
ODE	Open Drain Output Enable This bit controls output driver configuration for the pads connected to this signal. Either open drain or push/pull driver configurations can be selected. This feature applies to output pads only. 0: Open drain enable signal negated for the pad 1: Open drain enable signal asserted for the pad
SRC	Slew Rate Control 0: Slowest configuration 1: Fastest configuration
WPE	Weak Pull Up/Down Enable This bit controls whether the weak pull up/down devices are enabled/disabled for the pad connected to this signal. 0: Weak pull device enable signal negated for the pad 1: Weak pull device enable signal asserted for the pad
WPS	Weak Pull Up/Down Select This bit controls whether weak pull up or weak pull down devices are used for the pads connected to this signal when weak pull up/down devices are enabled. 0: Pull down enabled 1: Pull up enabled

Table 102. PCR[n] reset value exceptions

Field	Description
PCR[36] PCR[37] PCR[38]	These registers correspond to the ABS[0], ABS[2], and FAB boot pins, respectively. Their default state is input, pull enabled. Their reset value is 0x0102.
	This register corresponds to the TDO pin. Its default state is ALT1, slew rate = 1. Its reset value is 0x0604.
	This register corresponds to the TDI pin. Its default state is input, pull enabled, pull selected, slew enabled. So its reset value is 0x0107.
PCR[n]	For other PCR[n] registers, the reset value is 0x0000.

13.5.3.9 Pad Selection for Multiplexed Inputs Registers (PSMI0–PSMI25)

Figure 105 is the generic figure for PSMI register set. To see actual implementation, refer to Table 104.

Via routing it is possible to define different pads to be possible inputs for a certain peripheral function.

Address: Base + SIUL address offset¹ Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PADSEL0				0	0	0	0	PADSEL1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PADSEL2				0	0	0	0	PADSEL3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 105. Pad Selection for Multiplexed Inputs Register (PSMI)

¹ For SIUL address offset, refer to Table 104.

Table 103. PSMI field descriptions

Field	Description
PADSEL0–3, PADSEL4–7, ...	Pad Selection Bits Each PADSEL field selects the pad currently used for a certain input function. See Table 104.

In order to multiplex different pads to the same peripheral input, the SIUL provides a register that controls the selection between the different sources.

Table 104. Peripheral input pin selection

PSMI registers	PADSEL fields	SIUL address offset	Function / Peripheral	Mapping ¹
PSMI0	PADSEL35	0x500	CAN0RX	10: PCR[35]
	PADSEL12			01: PCR[12]
	PADSEL17			00: PCR[17]
	PADSEL57			11: PCR[57]
PSMI1	PADSEL15	0x501	DSPI0	0: PCR[15]
	PADSEL69			1: PCR[69]
PSMI2	PADSEL10	0x502	DSPI 0	000: PCR[10]
	PADSEL14			001: PCR[14]
	PADSEL 41			010: PCR[41]
	PADSEL 60			011: PCR[60]
	PADSEL 70			100: PCR[70]
PSMI3	PADSEL12	0x503	DSPI 0	00: PCR[12]
	PADSEL 13			01: PCR[13]
	PADSEL 68			10: PCR[68]
PSMI4	PADSEL 2	0x504	DSPI 1	00: PCR[2]
	PADSEL 69			01: PCR[69]
	PADSEL 15			10: PCR[15]
PSMI5	PADSEL 0	0x505	DSPI 1	00: PCR[0]
	PADSEL 42			01: PCR[42]
	PADSEL 70			10: PCR[70]
PSMI6	PADSEL 8	0x506	DSPI 1	00: PCR[8]
	PADSEL 11			01: PCR[11]
	PADSEL 38			10: PCR[38]
	PADSEL 68			11: PCR[68]
PSMI7	PADSEL 5	0x507	DSPI 2	0: PCR[5]
	PADSEL 69			1: PCR[69]
PSMI8	PADSEL 3	0x508	DSPI 2	0: PCR[3]
	PADSEL 70			1: PCR[70]
PSMI9	PADSEL 6	0x509	DSPI 2	0: PCR[6]
	PADSEL 68			1: PCR[68]

Table 104. Peripheral input pin selection (continued)

PSMI registers	PADSEL fields	SIUL address offset	Function / Peripheral	Mapping ¹
PSMI10	PADSEL 34	0x50A	eTimer0	00: PCR[34]
	PADSEL 59			01: PCR[59]
	PADSEL 60			10: PCR[60]
	PADSEL 15			11: PCR[15]
PSMI11	PADSEL 35	0x50B	eTimer0	00: PCR[35]
	PADSEL 43			01: PCR[43]
	PADSEL 60			10: PCR[60]
	PADSEL 6			11: PCR[6]
PSMI12	PADSEL 19	0x50C	eTimer0	00: PCR[19]
	PADSEL 43			01: PCR[43]
	PADSEL 57			10: PCR[57]
	PADSEL 7			11: PCR[7]
PSMI13	PADSEL 37	0x50D	eTimer0	00: PCR[37]
	PADSEL 43			01: PCR[43]
	PADSEL 57			10: PCR[57]
	PADSEL 4			11: PCR[4]
PSMI14	PADSEL 36	0x50E	eTimer0	00: PCR[36]
	PADSEL 42			01: PCR[42]
	PADSEL 59			10: PCR[59]
	PADSEL 5			11: PCR[5]
PSMI15	PADSEL 10	0x50F	eTimer0	00: PCR[10]
	PADSEL 42			01: PCR[42]
	PADSEL 59			10: PCR[59]
	PADSEL 2			11: PCR[2]
PSMI16	PADSEL 34	0x510	LIN0	00: PCR[19]
	PADSEL 11			01: PCR[11]
	PADSEL 19			10: PCR[34]
	PADSEL 40			11: PCR[40]
PSMI17	PADSEL 8	0x511	LIN1	00: PCR[8]
	PADSEL 11			01: PCR[11]
	PADSEL 39			10: PCR[39]
	PADSEL 66			11: PCR[66]

Table 104. Peripheral input pin selection (continued)

PSMI registers	PADSEL fields	SIUL address offset	Function / Peripheral	Mapping ¹
PSMI18	PADSEL 18	0x512	RTC	00: PCR[18]
	PADSEL 36			01: PCR[36]
	PADSEL 44			10: PCR[44]
PSMI19	PADSEL 17	0x513	RTC	0: PCR[17]
	PADSEL 44			1: PCR[44]
PSMI20	PADSEL 8	0x514	SAI1 BCLK	0: PCR[8]
	PADSEL 16			1: PCR[16]
PSMI21	PADSEL 2	0x515	SAI1 RXDATA	00: PCR[2]
	PADSEL 5			01: PCR[5]
	PADSEL 17			10: PCR[17]
PSMI22	PADSEL 5	0x516	SAI1	0: PCR[5]
	PADSEL 35			1: PCR[35]
PSMI23	PADSEL 3	0x517	SAI2	0: PCR[3]
	PADSEL 8			1: PCR[8]
PSMI24	PADSEL 6	0x518	Video	00: PCR[6]
	PADSEL 34			01: PCR[34]
	PADSEL 46			10: PCR[46]
PSMI25	PADSEL 7	0x519	Video	00: PCR[7]
	PADSEL 35			01: PCR[35]
	PADSEL 45			10: PCR[45]

¹ See the signal description chapter of this reference manual for correspondence between PCR and pinout

13.5.3.10 GPIO Pad Data Output Registers (GPDO0_3–GPDO68_71)

These registers are used to set or clear GPIO pads. Each pad data out bit can be controlled separately with a byte access.

Address: Base + (0x0600–0x0644) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PDO [2]	0	0	0	0	0	0	0	PDO [3]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 106. Port GPIO Pad Data Output Register 0–3 (GPDO0_3)

Table 105. GPDO0_3 field descriptions

Field	Description
PDO[x]	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0: Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1: Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

13.5.3.11 GPIO Pad Data Input Registers (GPDI0_3–GPDI68_71)

These registers are used to read the GPIO pad data with a byte access.

Address: Base + (0x0800–0x0844) Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	PDI [2]	0	0	0	0	0	0	0	PDI [3]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 107. Port GPIO Pad Data Input Register 0–3 (GPDI0_3)

Table 106. GPDO0_3 field descriptions

Field	Description
PDI[x]	Pad Data In This bit stores the value of the external GPIO pad associated with this register. 0: Value of the data in signal for the corresponding GPIO pad is logic low 1: Value of the data in signal for the corresponding GPIO pad is logic high

13.5.3.12 Parallel GPIO Pad Data Out Registers (PGPDO0 – PGPDO2)

MPC5604E devices ports are constructed such that they contain 16 GPIO pins, for example PortA[0..15]. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration.

Writing a parallel PGPDO register directly sets the associated GPDO register bits. There is also a masked parallel port output register allowing the user to determine which pins within a port are written.

While very convenient and fast, this approach does have implications regarding current consumption for the device power segment containing the port GPIO pads. Toggling several GPIO pins simultaneously can significantly increase current consumption.

WARNING

Caution must be taken to avoid exceeding maximum current thresholds when toggling multiple GPIO pins simultaneously. Please refer to data sheet.

Table 107 shows the locations and structure of the PGPDO_x registers.

Table 107. PGPDO0 – PGPDO6 Register Map

Offset ¹	Register	Field																															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0C00	PGPDO0	Port A															Port B																
0x0C04	PGPDO1	Port C															Port D																
0x0C08	PGPDO2	Port E								Reserved																							

¹ SIU base address is 0x3F9_0000. To calculate register address add offset to base address

It is important to note the bit ordering of the ports in the parallel port registers. The most significant bit of the parallel port register corresponds to the least significant pin in the port.

For example in Table 107, the PGPDO0 register contains fields for Port A and Port B.

- Bit 0 is mapped to Port A[0], bit 1 is mapped to Port A[1] and so on, through bit 15, which is mapped to Port A[15]
- Bit 16 is mapped to Port B[0], bit 17 is mapped to Port B[1] and so on, through bit 31, which is mapped to Port B[15].

13.5.3.13 Parallel GPIO Pad Data In Register (PGPDI0 – PGPDI2)

The SIU_PGPDI registers are similar in operation to the PGPDI registers, described in the previous section (Section 13.5.3.12, “Parallel GPIO Pad Data Out Registers (PGPDO0 – PGPDO2)”) but they are used to read port pins simultaneously.

NOTE

The port pins to be read need to be configured as inputs but even if a single pin within a port has IBE set, then you can still read that pin using the parallel port register. However, this does mean you need to be very careful.

Reads of PGPDI registers are equivalent to reading the corresponding GPD registers but significantly faster since as many as two ports can be read simultaneously with a single 32-bit read operation.

Table 108 shows the locations and structure of the PGPDI registers. Each 32-bit PGPDI register contains two 16-bit fields, each field containing the values for a separate port.

Table 108. PGPDI0 – PGPDI6 Register Map

Offset ¹	Register	Field																															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0C40	PGPDI0	Port A																Port B															
0x0C44	PGPDI1	Port C																Port D															
0x0C48	PGPDI2	Port E								Reserved																							

¹ SIU base address is 0xC3F9_0000. To calculate register address add offset to base address

It is important to note the bit ordering of the ports in the parallel port registers. The most significant bit of the parallel port register corresponds to the least significant pin in the port.

For example in Table 108, the PGPDI0 register contains fields for Port A and Port B.

- Bit 0 is mapped to Port A[0], bit 1 is mapped to Port A[1] and so on, through bit 15, which is mapped to Port A[15]
- Bit 16 is mapped to Port B[0], bit 17 is mapped to Port B[1] and so on, through bit 31, which is mapped to Port B[15].

13.5.3.14 Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO4)

The MPGPDO_x registers are similar in operation to the PGPDO_x ports described in Section 13.5.3.12, “Parallel GPIO Pad Data Out Registers (PGPDO0 – PGPDO2)”, but with two significant differences:

- The MPGPDO_x registers support *masked* port-wide changes to the data out on the pads of the respective port. Masking effectively allows selective bitwise writes to the full 16-bit port.
- Each 32-bit MPGPDO_x register is associated to only one port.

NOTE

The MPGPDO_x registers may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and will cause a transfer error response by the module. Read accesses return ‘0’.

Table 109 shows the locations and structure of the MPGPDO_x registers. Each 32-bit MPGPDO_x register contains two 16-bit fields (MASK_x and MPPDO_x). The MASK field is a bitwise mask for its associated port. The MPPDO0 field contains the data to be written to the port.

Table 109. MPGPDO0 – MPGPDO4 Register Map

Offset ¹	Register	Field																															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0C80	MPGPDO0	MASK0 (Port A)																MPPDO0 (Port A)															
0x0C84	MPGPDO1	MASK1 (Port B)																MPPDO1 (Port B)															
0x0C88	MPGPDO2	MASK2 (Port C)																MPPDO2 (Port C)															
0x0C8C	MPGPDO3	MASK3 (Port D)																MPPDO3 (Port D)															
0x0C90	MPGPDO4	MASK4 (Port E)								Reserved								MPPDO4 (Port E)															

¹ SIU base address is 0xC3F9_0000. To calculate register address add offset to base address

It is important to note the bit ordering of the ports in the parallel port registers. The most significant bit of the parallel port register corresponds to the least significant pin in the port.

For example in Table 109, the MPGPDO0 register contains field MASK0, which is the bitwise mask for Port A and field MPPDO0, which contains data to be written to Port A.

- MPGPDO0[0] is the mask bit for Port A[0], MPGPDO0[1] is the mask bit for Port A[1] and so on, through MPGPDO0[15], which is the mask bit for Port A[15]
- MPGPDO0[16] is the data bit mapped to Port A[0], MPGPDO0[17] is mapped to Port A[1] and so on, through MPGPDO0[31], which is mapped to Port A[15].

Table 110. MPGPDO0..MPGPDO4 field descriptions

Field	Description
MASK _x [15:0]	Mask Field Each bit corresponds to one data bit in the MPPDO _x register at the same bit location. 0: Associated bit value in the MPPDO _x field is ignored 1: Associated bit value in the MPPDO _x field is written
MPPDO _x [15:0]	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Accesses to this register location are coherent with accesses to the bitwise GPIO Pad Data Output Registers (GPDO0_3–GPDO68_71). The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

13.5.3.15 Interrupt Filter Maximum Counter Registers (IFMC0–IFMC31)

These registers are used to configure the filter counter associated with each digital glitch filter.

NOTE

For the pad transition to trigger an interrupt it must be steady for at least the filter period.

Address: Base + (0x1000–0x107C) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MAXCNTx[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 108. Interrupt Filter Maximum Counter Registers (IFMC0–IFMC31)

Table 111. IFMC field descriptions

Field	Description
MAXCNTx	Maximum Interrupt Filter Counter setting. Filter Period = T(CK)*3 (for 2 < MAXCNT < 6) Filter Period = T(CK)*MAXCNTx (for MAXCNT = 6,7,.... 15) For MAXCNT = 0, 1, 2 the filter behaves as ALL PASS filter. MAXCNTx can be 0 to 15; T(CK): Prescaled Filter Clock Period, which is IRC clock prescaled to IFCP value; T(IRC): Basic Filter Clock Period: 62.5 ns (F = 16 MHz).

13.5.3.16 Interrupt Filter Clock Prescaler Register (IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filter counters in the SIUL.

Address: Base + 0x1080 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	IFCP[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 109. Interrupt Filter Clock Prescaler Register (IFCPR)

Table 112. IFCPR field descriptions

Field	Description
IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(FIRC) x (IFCP + 1) T(FIRC) is the fast internal RC oscillator period. IFCP can be 0 to 15.

13.6 Functional description

13.6.1 Pad control

The SIUL controls the configuration and electrical characteristic of the device pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The pad configuration registers (PCR_n, see [Section 13.5.3.8, “Pad Configuration Registers \(PCR\[0:70\]\)”](#)) allow software control of the static electrical characteristics of external pins with a single write. These are used to configure the following pad features:

- Open drain output enable
- Slew rate control
- Pull control
- Pad assignment
- Control of analog path switches
- Safe mode behavior configuration

13.6.2 General purpose input and output pads (GPIO)

The SIUL manages up to 199 GPIO pads organized as ports that can be accessed for data reads and writes as 32, 16 or 8-bit.¹

1. There are exceptions. Some pads, e.g., precision analog pads, are input only.

As shown in [Figure 110](#), all port accesses are identical with each read or write being performed only at a different location to access a different port width.

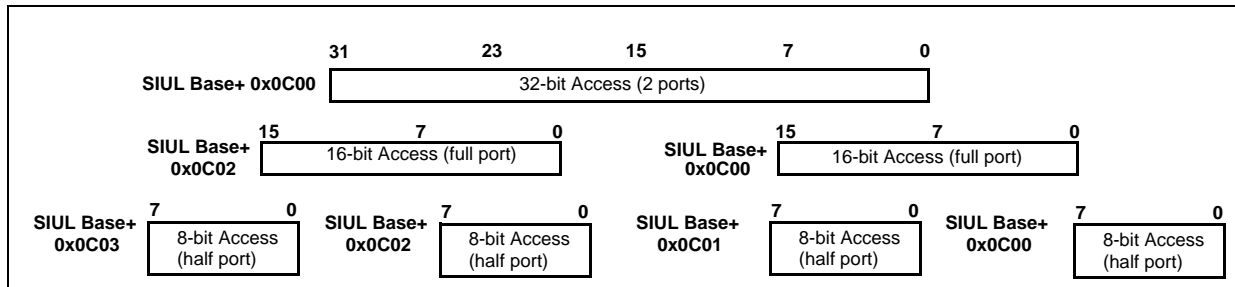


Figure 110. Data Port example arrangement showing configuration for different port width accesses

The SIUL has separate data input (GPDIn_n, see [Section 13.5.3.11, “GPIO Pad Data Input Registers \(GPDI0_3–GPDI68_71\)”](#)) and data output (GPDO_n, see [Section 13.5.3.10, “GPIO Pad Data Output Registers \(GPDO0_3–GPDO68_71\)”](#)) registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

Data output registers allow an output pad to be driven high or low (with the option of push-pull or open drain drive). Input registers are read-only and reflect the respective pad value. When the pad is configured to use one of its alternate functions, the data input value reflects the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non-GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

The allocation of what input function is connected to the pin is defined by the PSMI registers (PCR_n, see [Section 13.5.3.9, “Pad Selection for Multiplexed Inputs Registers \(PSMI0–PSMI25\)”](#))

13.6.3 External interrupts

The SIUL supports 24 external interrupts, EIRQ0–EIRQ23. In the signal description chapter of this reference manual, mapping is shown for external interrupts to pads.

The SIUL supports three interrupt vectors to the interrupt controller. Each vector interrupt has eight external interrupts combined together with the presence of flag generating an interrupt for that vector if enabled. All of the external interrupt pads within a single group have equal priority.

Refer to [Figure 111](#) for an overview of the external interrupt implementation.

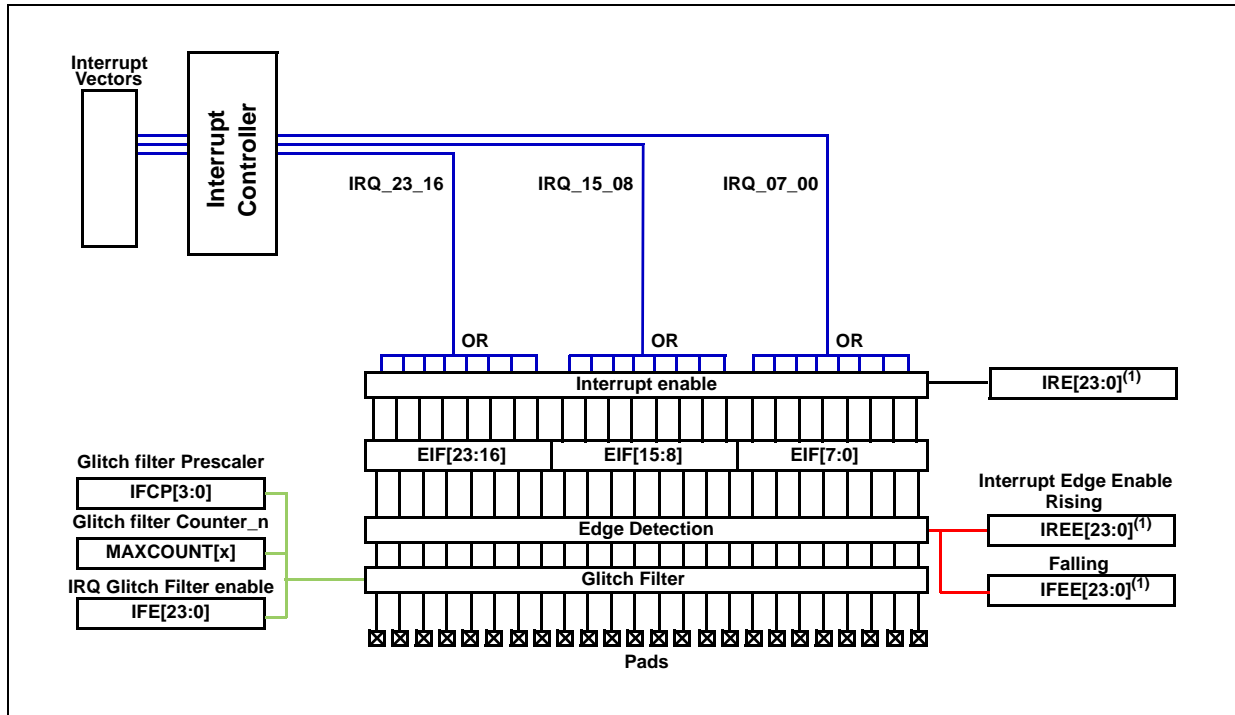


Figure 111. External interrupt pad diagram

Each interrupt can be enabled or disabled independently. This can be performed using the [Interrupt Request Enable Register \(IRER\)](#). A pad defined as an external interrupt can be configured to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured.

The active EIRQ edge is controlled through the configuration of the registers IREER and IFEEER.

Each external interrupt supports an individual flag which is held in the Interrupt Status Flag Register (ISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

13.7 Pin muxing

For pin muxing, please refer to the signal description chapter of this reference manual.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 14

e200z0h Core

14.1 Overview

The MPC5604E microcontroller implements the e200z0h core.

The e200 processor family is a set of CPU cores that implement low-cost versions of the Power Architecture™ Book E architecture. e200 processors are designed for deeply embedded control applications that require low cost solutions rather than maximum performance.

The e200z0h processors integrate an integer execution unit, branch control unit, instruction fetch and load/store units, and a multi-ported register file capable of sustaining three read and two write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in some cases.

The e200z0 core is a single-issue, 32-bit Power Architecture Book E VLE-only design with 32-bit general purpose registers (GPRs). All arithmetic instructions that execute in the core operate on data in the GPRs.

Instead of the base Power Architecture Book E instruction set support, the e200z0 core only implements the VLE (variable-length encoding) APU, providing improved code density. The VLE APU is further documented in the *PowerPC™ VLE APU Definition*, a separate document.

14.2 Features

The following is a list of some of the key features of the e200z0h core:

- High performance e200z0 core processor for managing peripherals and interrupts
- 32-bit Power Architecture Book E VLE-only programmer's model
- Single issue, 32-bit CPU
- Implements the VLE APU for reduced code footprint
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
 - Dedicated branch address calculation adder
 - Branch acceleration using Branch Target Buffer (e200z0h only)
- Supports independent instruction and data accesses to different memory subsystems, such as SRAM and Flash memory via independent Instruction and Data bus interface units (BIUs)
- Load/store unit
 - 1 cycle load latency

- Fully pipelined
- Big-endian support only
- Misaligned access support
- Zero load-to-use pipeline bubbles for aligned transfers
- Power management
 - Low power design
 - Dynamic power management of execution units
- Testability
 - Synthesizable, full MuxD scan design
 - ABIST/MBIST for optional memory arrays

14.2.1 Microarchitecture summary

The e200z0 processor utilizes a four stage pipeline for instruction execution. The Instruction Fetch (stage 1), Instruction Decode/Register file Read/Effective Address Calculation (stage 2), Execute/Memory Access (stage 3), and Register Writeback (stage 4) stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution unit consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), an 8×32 Hardware Multiplier array, result feed-forward hardware, and a hardware divider.

Arithmetic and logical operations are executed in a single cycle with the exception of the divide and multiply instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incremter and a dedicated Branch Address adder to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching from the BTB is performed to accelerate certain taken branches. Prefetched instructions are placed into an instruction buffer with 4 entries, each capable of holding a single 32-bit instruction or a pair of 16-bit instructions.

Conditional branches that are not taken execute in a single clock. Branches with successful target prefetching have an effective execution time of one clock on e200z0h. All other taken branches have an execution time of two clocks.

Memory load and store operations are provided for byte, halfword, and word (32-bit) data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized. Also, a load-to-use dependency does not incur any pipeline bubbles for most cases.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power Architecture architecture. The condition register consists of eight 4-bit fields that reflect the

results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

14.2.1.1 Block diagram

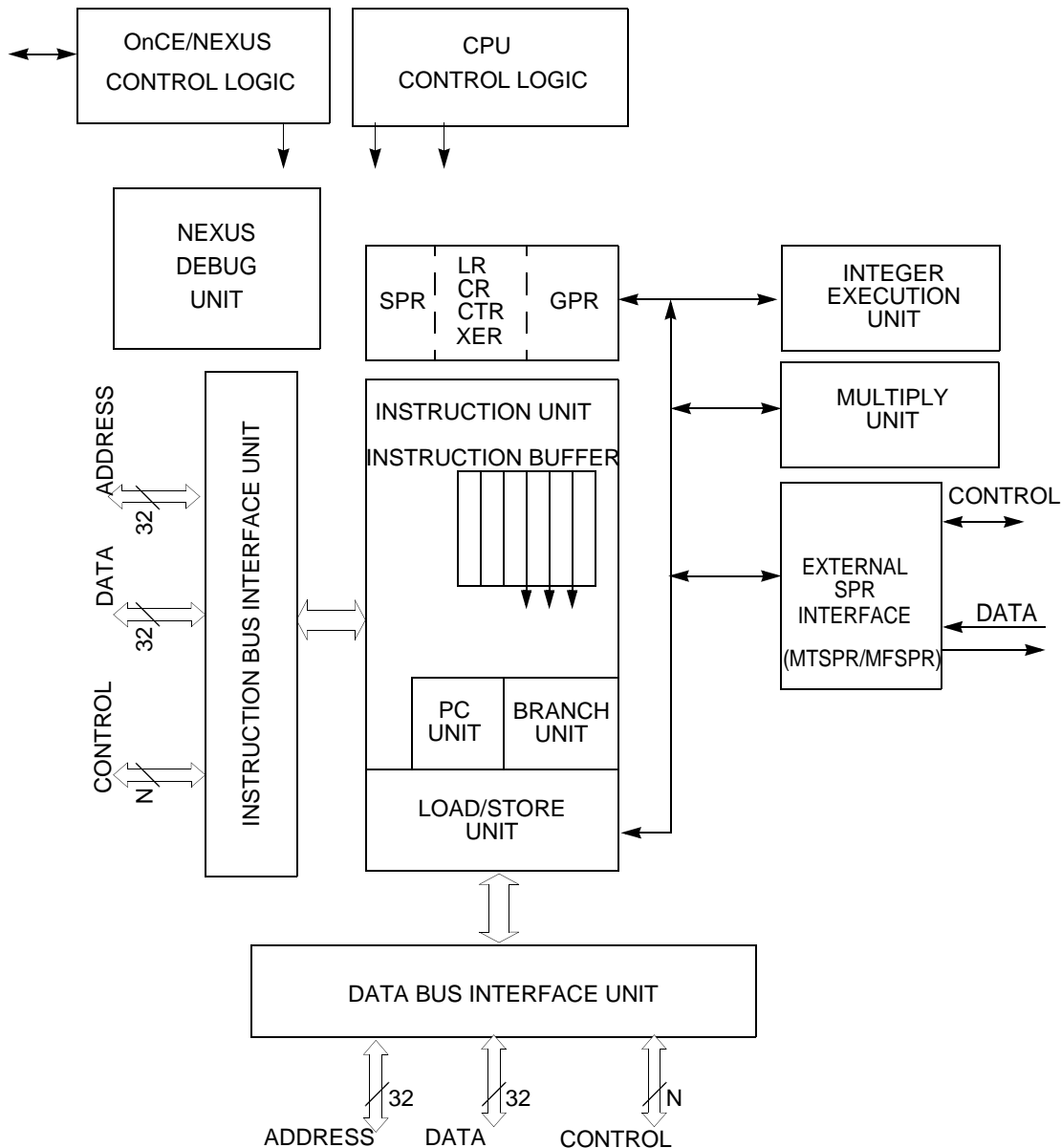


Figure 112. e200z0h block diagram

14.2.1.2 Instruction unit features

The features of the e200 Instruction unit are:

- 32-bit instruction fetch path supports fetching of one 32-bit instruction per clock, or as many as two 16-bit VLE instructions per clock
- Instruction buffer with 4 entries in e200z0h, each holding a single 32-bit instruction, or a pair of 16-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder supporting single cycle of execution of certain branches, two cycles for all others

14.2.1.3 Integer unit features

The e200 integer unit supports single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for count leading zero's function
- 32-bit single cycle barrel shifter for shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 5 to 34 clocks with minimized execution timing
- 8×32 hardware multiplier array supports 1 to 4 cycle $32 \times 32 \rightarrow 32$ multiply (early out)

14.2.1.4 Load/Store unit features

The e200 load/store unit supports load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- 32-bit dedicated interface to memory

14.2.1.5 e200z0h system bus features

The features of the e200z0h System Bus interface are as follows:

- Independent Instruction and Data Buses
- AMBA AHB Lite Rev 2.0 Specification with support for ARM v6 AMBA Extensions
 - Exclusive Access Monitor
 - Byte Lane Strobes
 - Cache Allocate Support
- 32-bit address bus plus attributes and control on each bus
- 32-bit read data bus for Instruction Interface
- Separate uni-directional 32-bit read data bus and 32-bit write data bus for Data Interface
- Overlapped, in-order accesses

14.2.1.6 Nexus features

The Nexus 2+ module is compliant with Class 2 of the IEEE-ISTO 5001-2003 standard, with additional Class 3 and Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus, static code may be traced.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to the processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary interface
- Watchpoint Trigger enable of Program Trace Messaging
- Auxiliary interface for higher data input/output
 - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
 - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
 - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
 - One (1) Watchpoint Event pin (**nex_evto_b**)
 - One (1) Event In pin (**nex_evti_b**)
 - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Ownership Trace and Watchpoint Trigger control
- All features controllable and configurable via the JTAG port

14.3 Core registers and programmer's model

This section describes the registers implemented in the e200z0h core. It includes an overview of registers defined by the Power Architecture Book E architecture, highlighting differences in how these registers are implemented in the e200 core, and provides a detailed description of e200-specific registers. Full descriptions of the architecture-defined register set are provided in Power Architecture Book E Specification.

The Power Architecture Book E defines register-to-register operations for all computational instructions. Source data for these instructions are accessed from the on-chip registers or are provided as immediate values embedded in the opcode. The three-register instruction format allows specification of a target register distinct from the two source registers, thus preserving the original data for use by other instructions. Data is transferred between memory and registers with explicit load and store instructions only.

Figure 113 and Figure 114 show the e200 register set, including the registers that are accessible while in supervisor mode and the registers that are accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

NOTE

e200z0h is a 32-bit implementation of the Power Architecture Book E specification. In this document, register bits are sometimes numbered from bit 0 (Most Significant Bit) to 31 (Least Significant Bit), rather than the Book E numbering scheme of 32:63, thus register bit numbers for some registers in Book E are 32 higher.

Where appropriate, the Book E defined bit numbers are shown in parentheses.

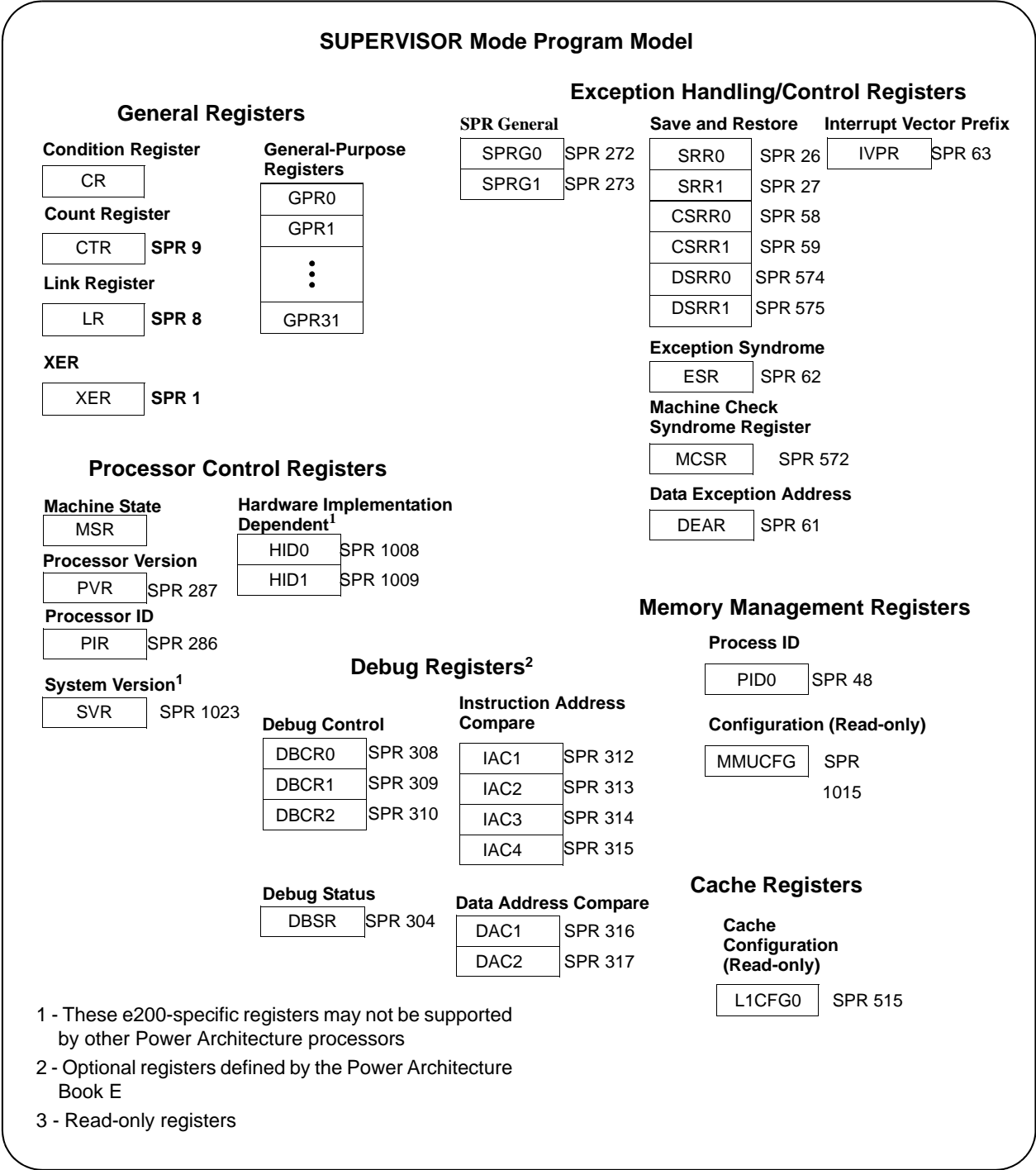


Figure 113. e200z0 Supervisor mode programmer's model

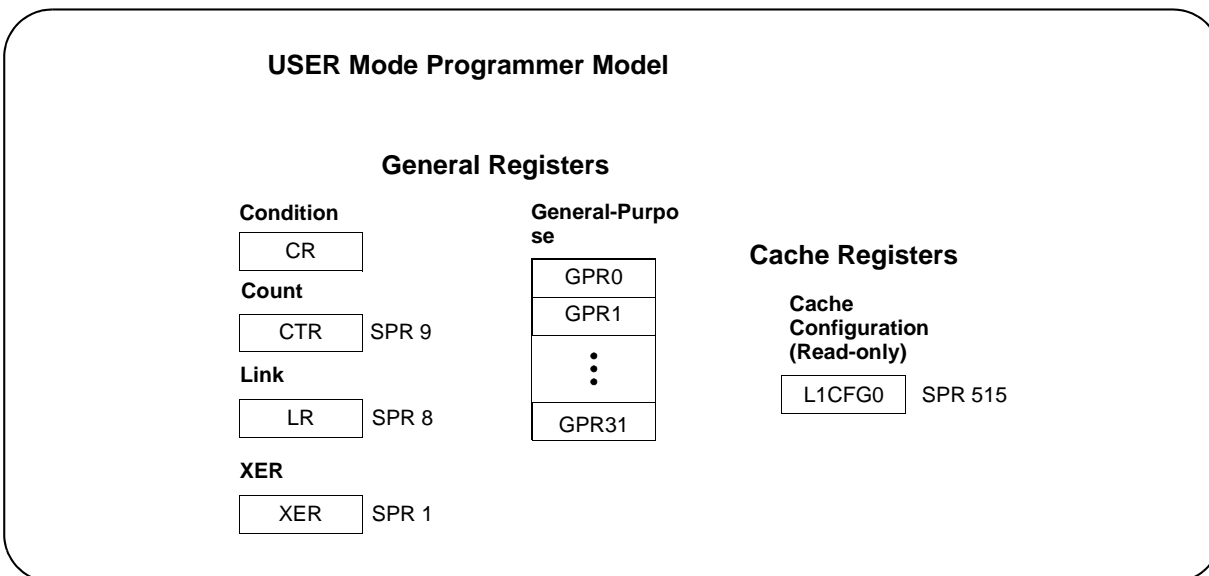


Figure 114. e200 User mode program model

14.3.1 Unimplemented SPRs and read-only SPRs

e200 fully decodes the SPR field of the **mf spr** and **mt spr** instructions. If the SPR specified is undefined and not privileged, an illegal instruction exception is generated. If the SPR specified is undefined and privileged and the CPU is in user mode (MSR[PR=1]), a privileged instruction exception is generated. If the SPR specified is undefined and privileged and the core is in supervisor mode (MSR[PR=0]), an illegal instruction exception is generated.

For the **mt spr** instruction, if the SPR specified is read-only and not privileged, an illegal instruction exception is generated. If the SPR specified is read-only and privileged and the core is in user mode (MSR[PR=1]), a privileged instruction exception is generated. If the SPR specified is read-only and privileged and the core is in supervisor mode (MSR[PR=0]), an illegal instruction exception is generated.

14.4 Instruction summary

The e200z0 core supports all VLE instructions described in the *PowerPC™ VLE APU Definition version 1.2* together with the additional instructions for context save/restore.

Chapter 15

Crossbar Switch (XBAR)

15.1 Introduction

This chapter describes the multi-port crossbar switch (XBAR), which supports simultaneous connections between four master ports and four slave ports. XBAR supports a 32-bit address bus width and a 32-bit data bus width at all master and slave ports.

15.2 Block diagram

Figure 115 shows a block diagram of the crossbar switch.

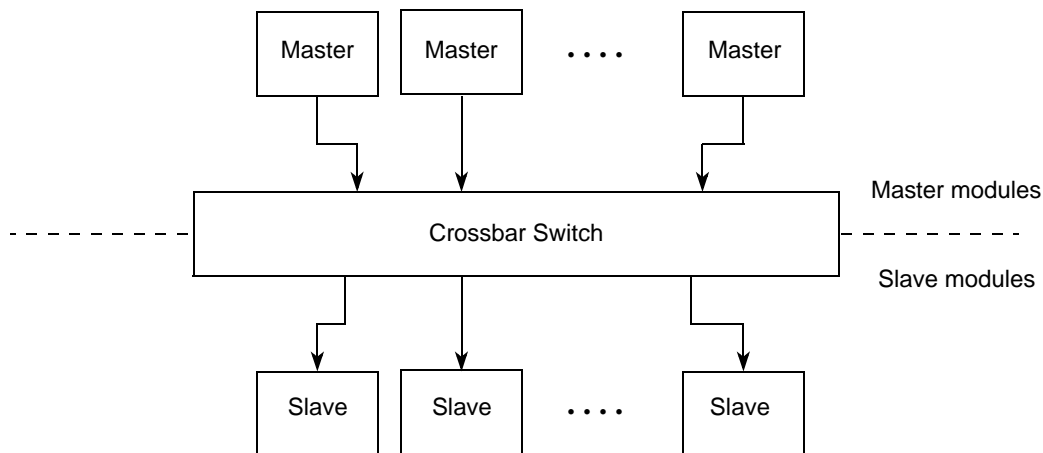


Figure 115. XBAR block diagram

Table 113 gives the crossbar switch port for each master and slave, the assigned and fixed ID number for each master and shows the master ID numbers as they relate to the master port numbers.

Table 113. Device XBAR switch ports

Module	Port		Physical master ID
	Type	Logical number	
e200z0 core—CPU instructions	Master	0	0
e200z0 core—Data	Master	1	1
eDMA	Master	2	2
Ethernet	Master		4
Flash	Slave	0	—
Internal SRAM	Slave	2	—
Video encoder output buffer	Slave		
Peripheral bridge	Slave	7	—

15.3 Overview

The XBAR allows for concurrent transactions to occur from any master port to any slave port. It is possible for all master ports and slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic selects the higher priority master and grants it ownership of the slave port. All other masters requesting that slave port are stalled until the higher priority master completes its transactions.

Requesting masters are granted access based on a fixed priority.

15.4 Features

- 4 Master ports
 - e200z0 core complex Instruction port
 - e200z0 core complex Load/Store Data port
 - eDMA
 - Ethernet
- 4 Slave ports
 - Flash memory (code flash and data flash) controller
 - SRAM controller
 - Video encoder output buffer
 - Peripheral bridge
- 32-bit internal address, 32-bit internal data paths
- Fully concurrent transfers between independent master and slave ports
- Fixed priority scheme and fixed parking strategy

15.5 Modes of operation

15.5.1 Normal mode

In normal mode, the XBAR provides the register interface and logic that controls crossbar switch configuration.

15.5.2 Debug mode

The XBAR operation in debug mode is identical to operation in normal mode.

15.6 Functional description

This section describes the functionality of the XBAR in more detail.

15.6.1 Overview

The main goal of the XBAR is to increase overall system performance by allowing multiple masters to communicate concurrently with multiple slaves. To maximize data throughput, it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the XBAR stalls masters, or inserts bubbles on the slave side.

15.6.2 General operation

When a master makes an access to the XBAR from an idle master state, the access is taken immediately by the XBAR. If the targeted slave port of the access is available (that is, the requesting master is currently granted ownership of the slave port), the access is immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the XBAR by a granted master. If the targeted slave port of the access is busy or parked on a different master port, the requesting master receives wait states until the targeted slave port can service the master request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the XBAR appears to be just another slave to the master device, the master device has no indication that it owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it is wait-stated.

A master is given control of a targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has the following conditions:

- Outstanding request to slave port A that has a long response time
- Pending access to a different slave port B
- Lower priority master also makes a request to the different slave port B.

In this case, the lower priority master is granted bus ownership of slave port B after a cycle of arbitration, assuming the higher priority master slave port A access is not terminated.

After a master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle, leaves that slave port for its next access, or loses control of the slave port to a higher priority master with a request to the same slave port. However, because all masters run a fixed-length burst transfer to a slave port, it retains control of the slave port until that transfer sequence is completed.

When a slave bus is idled by the XBAR, it is parked on the master that did the last transfer.

15.6.3 Master ports

A master access is taken if the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case, the XBAR is completely transparent and the master access is immediately transmitted on the slave bus and no arbitration delays are incurred. A master access stalls if the access decodes to a slave port that is busy serving another master, parked on another master.

If the slave port is currently parked on another master, and no other master is requesting access to the slave port, then only one clock of arbitration is incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters, then the master gains control over the slave port as soon as the data phase of the current access is completed. If the slave port is currently servicing another master of a higher priority, then the master gains control of the slave port after the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

A master access is responded to with an error if the access decodes to a location not occupied by a slave port. This is the only time the XBAR directly responds with an error response. All other error responses received by the master are the result of error responses on the slave ports being passed through the XBAR.

15.6.4 Slave ports

The goal of the XBAR with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. To do this the XBAR must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the XBAR forces a bubble onto the slave bus when a master is actively making a request. This occurs when a handoff of bus ownership occurs and there are no wait states from the slave port. A requesting master that does not own the slave port is granted access after a one clock delay.

15.6.5 Priority assignment

Each master port is assigned a fixed 3-bit priority level (hard-wired priority). [Table 114](#) shows the priority levels assigned to each master (the lowest has highest priority).

Table 114. Hardwired bus master priorities

Module	Port		Priority level
	Type	Number	
e200z0 core—CPU instructions	Master	0	7
e200z0 core—Data	Master	1	6
eDMA	Master	2	5
Ethernet	Master		

15.6.6 Arbitration

XBAR supports only a fixed-priority comparison algorithm.

15.6.6.1 Fixed priority operation

When operating in fixed-priority arbitration mode, each master is assigned a unique priority level. If two masters both request access to a slave port, the master with the highest priority in the selected priority register gains control over the slave port.

Any time a master makes a request to a slave port, the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (if any). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the higher priority master is granted control at the termination of any currently pending access, assuming the pending transfer is not part of a burst transfer.

A new requesting master must wait until the end of the fixed-length burst transfer, before it is granted control of the slave port. But if the new requesting master's priority level is lower than that of the master that currently has control of the slave port, the new requesting master is forced to wait until the master that currently has control of the slave port is finished accessing the current slave port.

15.6.6.1.1 Parking

If no master is currently requesting the slave port, the slave port is parked. The slave port parks always to the most recently requesting master (park-on-last). When parked on the last master, the slave port is passing that master's signals through to the slave bus. When the master accesses the slave port again, no other arbitration penalties are incurred except that a one clock arbitration penalty is incurred for each access request to the slave port made by another master port. All other masters pay a one clock penalty.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 16

Miscellaneous Control Module (MCM)

16.1 Introduction

The Miscellaneous Control Module (MCM), provides miscellaneous control functions for the device Standard Product Platform (SPP) including program-visible information about the platform configuration and revision levels, a reset status register, a software watchdog timer, and wakeup control for exiting sleep modes, and optional features such as an address map for the device's crossbar switch, information on memory errors reported by error-correcting codes and/or generic access error information for certain processor cores. It also provides with register access protection for the following slave modules: INTC, MCM, STM, and SWT.

16.2 Overview

The Miscellaneous Control Module is mapped into the IPS space and supports a number of miscellaneous control functions for the platform device.

16.3 Features

The MCM includes these features:

- Program-visible information on the platform device configuration and revision
- registers for capturing information on platform memory errors if error-correcting codes (ECC) are implemented
- registers to specify the generation of single- and double-bit memory data inversions for test purposes if error-correcting codes are implemented
- access address information for faulted memory accesses for certain processor core micro-architectures,
- AXBS_lite priority functions, including forcing round robin and high priority enabling.
- Capability to restrict register access to supervisor mode to selected on-platform slave devices: INTC, MCM, STM, and SWT.

16.4 Memory Map and Registers Description

This section details the programming model for the Miscellaneous Control Module. This is an on-platform 128-byte space mapped to the region serviced by an IPS bus controller. Some of the control registers have a 64-bit width. These 64-bit registers are implemented as two 32-bit registers, and include an “H” and “L” suffixes, indicating the “high” and “low” portions of the control function.

Miscellaneous Control Module (MCM)

The Miscellaneous Control Module does not include any logic which provides access control. Rather, this function is supported using the standard access control logic provided by the IPS controller.

MCM registers are accessible only when the core is in supervisor mode (see [Section 16.4.3](#), “MCM_reg_protection”).

16.4.1 Memory Map

Table 115 is a 32-bit view of the MCM’s memory map.

Table 115. MCM 32-bit Memory Map

MCM Offset	Register			
0x0000	Processor Core Type (PCT)		Revision (REV)	
0x0008	IPS Module Configuration (IMC)			
0x000c	Reserved			
0x0014	Reserved			
0x0018	Reserved			
0x001c	Reserved		Misc Interrupt (MIR)	
0x0020	Reserved			
0x0024	Miscellaneous User-Defined Control Register (MUDCR)			
0x0028	Reserved			
0x002c - 0x003c	Reserved			
0x0040	Reserved		ECC Configuration (ECR)	
0x0044	Reserved		ECC Status (ESR)	
0x0048	Reserved		ECC Error Generation (EEGR)	
0x004c	Reserved			
0x0050	Flash ECC Address (FEAR)			
0x0054	Reserved		Flash ECC Master (FEMR)	Flash ECC Attributes (FEAT)
0x0058	Reserved			
0x005c	Flash ECC Data (FEDR)			
0x0060	RAM ECC Address (REAR)			
0x0064	Reserved	RAM ECC Syndrome (RESR)	RAM ECC Master (REMR)	RAM ECC Attributes (REAT)
0x0068	Reserved			
0x006c	RAM ECC Data (REDR)			

Table 115. MCM 32-bit Memory Map (continued)

MCM Offset	Register
0x0070 - 0x007c	Reserved

16.4.2 Registers Description

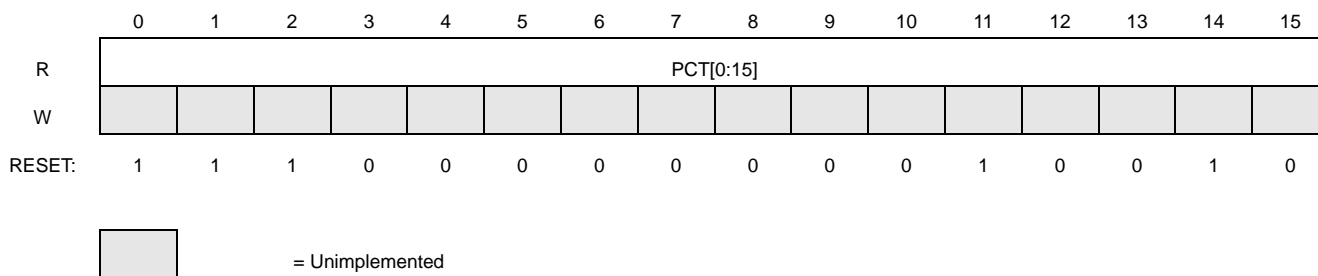
Attempted accesses to reserved addresses result in an error termination, while attempted writes to read-only registers are ignored and do not terminate with an error. Unless noted otherwise, writes to the programming model must match the size of the register, e.g., an n -bit register only supports n -bit writes, etc. Attempted writes of a different size than the register width produce an error termination of the bus cycle and no change to the targeted register.

16.4.2.1 Processor Core Type (PCT) register

The PCT is a 16-bit read-only register specifying the architecture of the processor core in the device. The state of this register is defined by a module input signal; it can only be read from the IPS programming model. Any attempted write is ignored.

See [Figure 116](#) and [Table 116](#) for the Processor Core Type definition.

Register address: MCM Base + 0x0000


Figure 116. Processor Core Type (PCT) Register

16.4.2.2 Revision (REV) register

The REV is a 16-bit read-only register specifying a revision number. The state of this register is defined by an input signal; it can only be read from the IPS programming model. Any attempted write is ignored.

See [Figure 117](#) and [Table 117](#) for the Revision definition.

Miscellaneous Control Module (MCM)

Register address: MCM Base + 0x0002

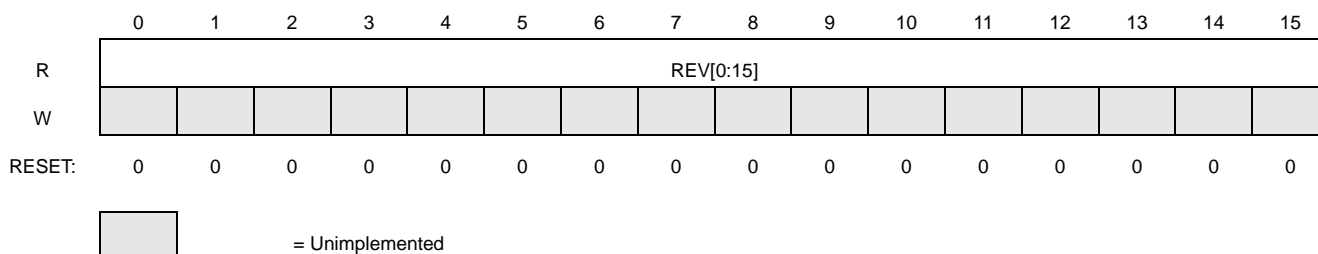


Figure 117. Revision (REV) Register

Table 117. Revision (REV) Field Descriptions

Name	Description
0-15 REV[0:15]	Revision The REV[0:15] field is specified by an input signal to define a software-visible revision number.

16.4.2.3 IPS Module Configuration (IMC) register

The IMC is a 32-bit read-only register identifying the presence/absence of the 32 low-order IPS peripheral modules connected to the primary IPI SkyBlue bus controller. The state of this register is defined by a module input signal; it can only be read from the IPS programming model. Any attempted write is ignored.

See [Figure 118](#) and [Table 118](#) for the IPS Module Configuration definition.

Register address: MCM Base + 0x0008

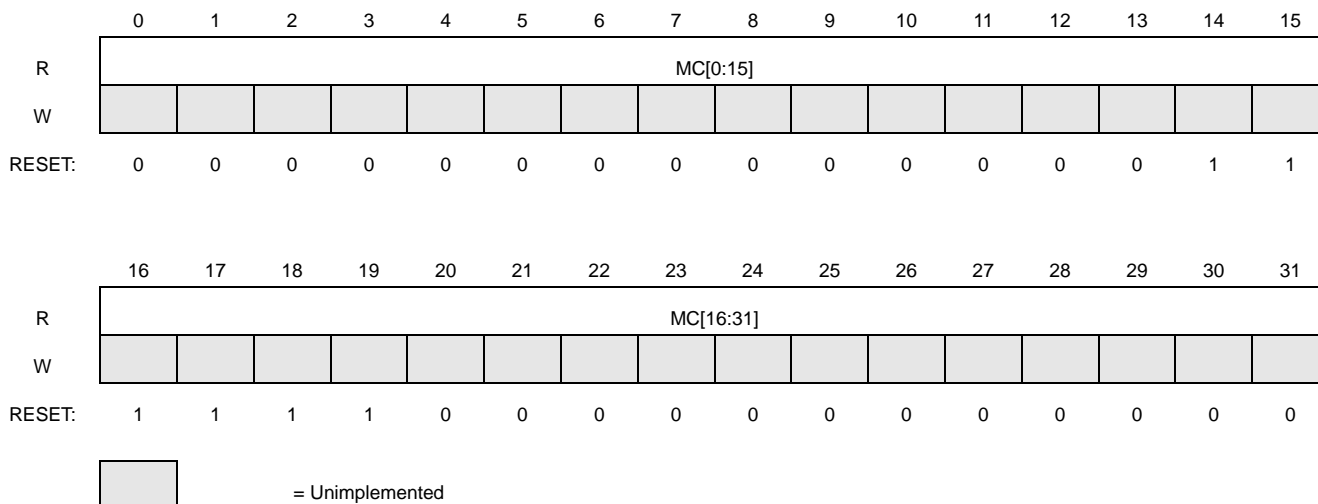


Figure 118. IPS Module Configuration (IMC) Register

Table 118. IPS Module Configuration (IMC) Field Descriptions

Name	Description
0-31 MC[0:31]	IPS Module Configuration MC[n] = 0 if an IPS module connection to decoded slot "n" is absent MC[n] = 1 if an IPS module connection to decoded slot "n" is present

16.4.2.4 Miscellaneous Interrupt Register (MIR)

All interrupt requests associated with MCM are collected in the MIR register. This includes the processor core system bus fault interrupt.

During the appropriate interrupt service routine handling these requests, the interrupt source contained in the MCMIR must be explicitly cleared. See [Figure 119](#) and [Table 119](#).

Register address: MCM Base + 0x001F

	0	1	2	3	4	5	6	7
R	FB0AI	FB0SI	FB1AI	FB1SI	0	0	0	0
W	1	1	1	1	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
RESET:	0	0	0	0	0	0	0	0

XXXXXXXX = Unimplemented

Figure 119. Miscellaneous Interrupt (MIR) Register

Table 119. Miscellaneous Interrupt (MIR) Field Descriptions

Name	Description
0 FB0AI	Flash Bank 0 Abort Interrupt 0: A flash bank 0 abort has not occurred. 1: A flash bank 0 abort has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.
1 FB0SI	Flash Bank 0 Stall Interrupt 0: A flash bank 0 stall has not occurred. 1: A flash bank 0 stall has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.
2 FB1AI	Flash Bank 1 Abort Interrupt 0: A flash bank 1 abort has not occurred. 1: A flash bank 1 abort has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.
3 FB1SI	Flash Bank 1 Stall Interrupt 0: A flash bank 1 stall has not occurred. 1: A flash bank 1 stall has occurred. The interrupt request is negated by writing a 1 to this bit. Writing a 0 has no effect.

16.4.2.5 Miscellaneous User-Defined Control Register (MUDCR)

The MUDCR provides a program-visible register for user-defined control functions. It typically is used as configuration control for miscellaneous SoC-level modules. The contents of this register is simply output from MCM to other modules where the user-defined control functions are implemented. See [Figure 120](#) and [Table 120](#) for the Miscellaneous User-Defined Control Register definition.

Miscellaneous Control Module (MCM)

Register address: MCM Base + 0x0024

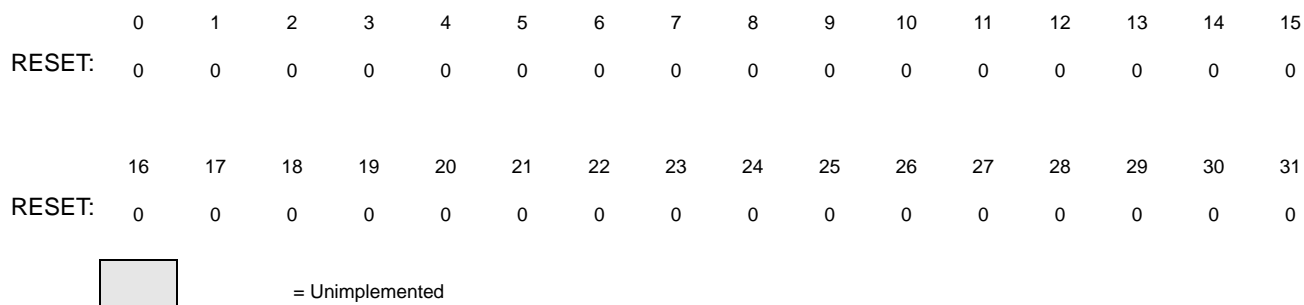


Figure 120. Miscellaneous User-Defined Control (MUDCR) Register

Table 120. Miscellaneous User-Defined Control Register (MUDCR) Field Descriptions

Name	Description
MUDCR	

AXBS_lite force_round_robin bit (MUDCR[31])-

When the AXBS_lite is included on the platform, this bit is used to drive the force_round_robin bit of the AXBS_lite. This will force the slaves into round robin mode of arbitration rather than fixed mode. Unless a master is using priority elevation, which forces the design back into fixed mode regardless of this bit. By defining the ‘define ENABLE_ROUND_ROBIN_RESET, this bit will reset to 1.

AXBS_lite is in round robin mode

AXBS_lite is in fixed priority mode

16.4.2.6 ECC registers

There are a number of program-visible registers for the sole purpose of reporting and logging of memory failures. These registers include the following:

- ECC Configuration Register (ECR)
- ECC Status Register (ESR)
- ECC Error Generation Register (EEGR)
- Flash ECC Address Register (FEAR)
- Flash ECC Master Number Register (FEMR)
- Flash ECC Attributes Register (FEAT)
- Flash ECC Data Register (FEDR)
- RAM ECC Address Register (REAR)
- RAM ECC Syndrome Register (RESR)
- RAM ECC Master Number Register (REMR)
- RAM ECC Attributes Register (REAT)
- RAM ECC Data Register (REDR)

The details on the ECC registers are provided in the subsequent sections. If the design does not include ECC on the memories, these addresses are reserved locations within the MCM's programming model.

16.4.2.7 ECC Configuration Register (ECR)

The ECC Configuration Register is an 8-bit control register for specifying which types of memory errors are reported. In all systems with ECC, the occurrence of a non-correctable error causes the current access to be terminated with an error condition. In many cases, this error termination is reported directly by the initiating bus master. However, there are certain situations where the occurrence of this type of non-correctable error is not reported by the master. Examples include speculative instruction fetches which are discarded due to a change-of-flow operation, and buffered operand writes. The ECC reporting logic in the MCM provides an optional error interrupt mechanism to signal all non-correctable memory errors. In addition to the interrupt generation, the MCM captures specific information (memory address, attributes and data, bus master number, etc.) which may be useful for subsequent failure analysis.

The reporting of single-bit memory corrections can only be enabled via a an SoC-configurable module input signal. While not directly accessible to a user, this capability is viewed as important for error logging and failure analysis.

See [Figure 121](#) and [Table 121](#) for the ECC Configuration Register definition.

Register address: MCM Base + 0x0043

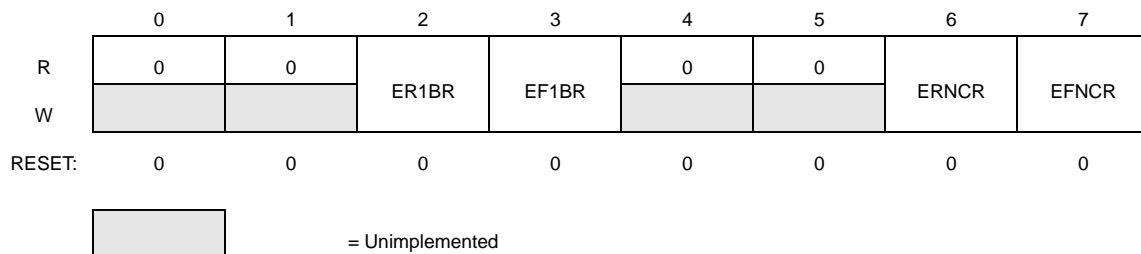


Figure 121. ECC Configuration (ECR) Register

Table 121. ECC Configuration (ECR) Field Definitions

Name	Description
2 ER1BR	<p>Enable RAM 1-bit Reporting 0 = Reporting of single-bit RAM corrections is disabled. 1 = Reporting of single-bit RAM corrections is enabled.</p> <p>This bit can only be set if the SoC-configurable input enable signal is asserted. The occurrence of a single-bit RAM correction generates a MCM ECC interrupt request as signalled by the assertion of ESR[R1BC]. The address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers.</p>
3 EF1BR	<p>Enable Flash 1-bit Reporting 0 = Reporting of single-bit flash corrections is disabled. 1 = Reporting of single-bit flash corrections is enabled.</p> <p>This bit can only be set if the SoC-configurable input enable signal is asserted. The occurrence of a single-bit flash correction generates a MCM ECC interrupt request as signalled by the assertion of ESR[F1BC]. The address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers.</p>

Table 121. ECC Configuration (ECR) Field Definitions

Name	Description
6 ERNCR	<p>Enable RAM Non-Correctable Reporting 0 = Reporting of non-correctable RAM errors is disabled. 1 = Reporting of non-correctable RAM errors is enabled.</p> <p>The occurrence of a non-correctable multi-bit RAM error generates a MCM ECC interrupt request as signalled by the assertion of ESR[RNCE]. The faulting address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers.</p>
7 EFNCR	<p>Enable Flash Non-Correctable Reporting 0 = Reporting of non-correctable flash errors is disabled. 1 = Reporting of non-correctable flash errors is enabled.</p> <p>The occurrence of a non-correctable multi-bit flash error generates a MCM ECC interrupt request as signalled by the assertion of ESR[FNCE]. The faulting address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers.</p>

16.4.2.8 ECC Status Register (ESR)

The ECC Status Register is an 8-bit control register for signaling which types of properly-enabled ECC events have been detected. The ESR signals the last, properly-enabled memory event to be detected. ECC interrupt generation is separated into single-bit error detection/correction, uncorrectable error detection and the combination of the two as defined by the following boolean equations:

```

MCM_ECC1BIT_IRQ
    = ECR[ER1BR] & ESR[R1BC]           // ram, 1-bit correction
    | ECR[EF1BR] & ESR[F1BC]           // flash, 1-bit correction
MCM_ECCRNCR_IRQ
    = ECR[ERNCR] & ESR[RNCE]           // ram, noncorrectable error
MCM_ECCFNCR_IRQ
    = ECR[EFNCR] & ESR[FNCE]           // flash, noncorrectable error
MCM_ECC2BIT_IRQ
    = MCM_ECCRNCR_IRQ                   // ram, noncorrectable error
    | MCM_ECCFNCR_IRQ                   // flash, noncorrectable error
MCM_ECC_IRQ
    = MCM_ECC1BIT_IRQ                   // 1-bit correction
    | MCM_ECC2BIT_IRQ                   // noncorrectable error
    
```

where the combination of a properly-enabled category in the ECR and the detection of the corresponding condition in the ESR produces the interrupt request.

The MCM allows a maximum of one bit of the ESR to be asserted at any given time. This preserves the association between the ESR and the corresponding address and attribute registers, which are loaded on each occurrence of an properly-enabled ECC event. If there is a pending ECC interrupt and another properly-enabled ECC event occurs, the MCM hardware automatically handles the ESR reporting, clearing the previous data and loading the new state and thus guaranteeing that only a single flag is asserted.

To maintain the coherent software view of the reported event, the following sequence in the MCM error interrupt service routine is suggested:

1. Read the ESR and save it.
2. Read and save all the address and attribute reporting registers.

3. Re-read the ESR and verify the current contents matches the original contents. If the two values are different, go back to step 1 and repeat.
4. When the values are identical, write a 1 to the asserted ESR flag to negate the interrupt request.

See [Figure 122](#) and [Table 122](#) for the ECC Status Register definition.

Register address: MCM Base + 0x0047

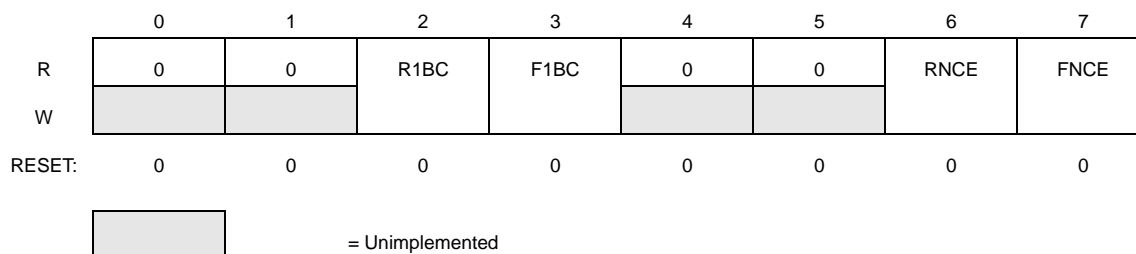


Figure 122. ECC Status (ESR) Register

Table 122. ECC Status (ESR) Field Definitions

Name	Description
2 R1BC	<p>RAM 1-bit Correction 0 = No reportable single-bit RAM correction has been detected. 1 = A reportable single-bit RAM correction has been detected.</p> <p>This bit can only be set if ECR[EP1BR] is asserted. The occurrence of a properly-enabled single-bit RAM correction generates a MCM ECC interrupt request. The address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.</p>
3 F1BC	<p>Flash 1-bit Correction 0 = No reportable single-bit flash correction has been detected. 1 = A reportable single-bit flash correction has been detected.</p> <p>This bit can only be set if ECR[EPF1BR] is asserted. The occurrence of a properly-enabled single-bit flash correction generates a MCM ECC interrupt request. The address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.</p>
6 RNCE	<p>RAM Non-Correctable Error 0 = No reportable non-correctable RAM error has been detected. 1 = A reportable non-correctable RAM error has been detected.</p> <p>The occurrence of a properly-enabled non-correctable RAM error generates a MCM ECC interrupt request. The faulting address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.</p>
7 FNCE	<p>Flash Non-Correctable Error 0 = No reportable non-correctable flash error has been detected. 1 = A reportable non-correctable flash error has been detected.</p> <p>The occurrence of a properly-enabled non-correctable flash error generates a MCM ECC interrupt request. The faulting address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect.</p>

In the event that multiple status flags are signaled simultaneously, MCM records the event with the R1BC as highest priority, then F1BC, then RNCE, and finally FNCE.

16.4.2.9 ECC Error Generation Register (EEGR)

The ECC Error Generation Register is a 16-bit control register used to force the generation of single- and double-bit data inversions in the memories with ECC, most notably the RAM. This capability is provided for two purposes:

- It provides a software-controlled mechanism for “injecting” errors into the memories during data writes to verify the integrity of the ECC logic.
- It provides a mechanism to allow testing of the software service routines associated with memory error logging.

It should be noted that while the EEGR is associated with the RAM, similar capabilities exist for the flash, i.e., the ability to program the non-volatile memory with single- or double-bit errors is supported for the same two reasons previously identified.

For both types of memories (RAM and flash), the intent is to generate errors during data write cycles, such that subsequent reads of the corrupted address locations generate ECC events, either single-bit corrections or double-bit noncorrectable errors that are terminated with an error response.

The enabling of these error generation modes requires the same SoC-configurable input enable signal (as that used to enable single-bit correction reporting) be asserted.

See [Figure 123](#) and [Table 123](#) for the ECC Configuration Register definition.

Register address: MCM Base + 0x004a

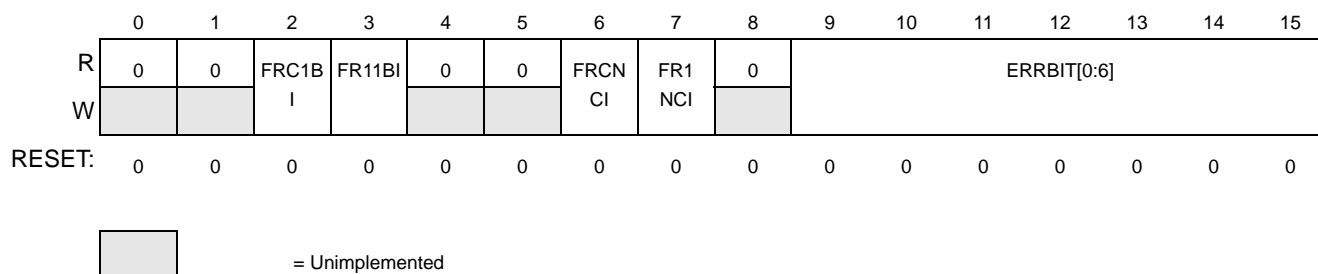


Figure 123. ECC Error Generation (EEGR) Register

Table 123. ECC Error Generation (EEGR) Field Definitions

Name	Description
2 FRC1BI	<p>Force RAM Continuous 1-Bit Data Inversions 0 = No RAM continuous 1-bit data inversions are generated. 1 = 1-bit data inversions in the RAM are continuously generated.</p> <p>The assertion of this bit forces the RAM controller to create 1-bit data inversions, as defined by the bit position specified in ERRBIT[0:6], continuously on every write operation.</p> <p>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM.</p> <p>After this bit has been enabled to generate another continuous 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic.</p> <p>This bit can only be set if the same SoC configurable input enable signal (as that used to enable single-bit correction reporting) is asserted.</p>
3 FR11BI	<p>Force RAM One 1-bit Data Inversion 0 = No RAM single 1-bit data inversion is generated. 1 = One 1-bit data inversion in the RAM is generated.</p> <p>The assertion of this bit forces the RAM controller to create one 1-bit data inversion, as defined by the bit position specified in ERRBIT[0:6], on the first write operation after this bit is set.</p> <p>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM.</p> <p>After this bit has been enabled to generate a single 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic.</p> <p>This bit can only be set if the same SoC configurable input enable signal (as that used to enable single-bit correction reporting) is asserted.</p>
6 FRCNCI	<p>Force RAM Continuous Noncorrectable Data Inversions 0 = No RAM continuous 2-bit data inversions are generated. 1 = 2-bit data inversions in the RAM are continuously generated.</p> <p>The assertion of this bit forces the RAM controller to create 2-bit data inversions, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, continuously on every write operation.</p> <p>After this bit has been enabled to generate another continuous noncorrectable data inversion, it must be cleared before being set again to properly re-enable the error generation logic.</p> <p>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM.</p>

Table 123. ECC Error Generation (EEGR) Field Definitions (continued)

Name	Description
<p>7 FR1NCI</p>	<p>Force RAM One Noncorrectable Data Inversions 0 = No RAM single 2-bit data inversions are generated. 1 = One 2-bit data inversion in the RAM is generated.</p> <p>The assertion of this bit forces the RAM controller to create one 2-bit data inversion, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, on the first write operation after this bit is set.</p> <p>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM.</p> <p>After this bit has been enabled to generate a single 2-bit error, it must be cleared before being set again to properly re-enable the error generation logic.</p>
<p>9-15 ERRBIT [0:6]</p>	<p>Error Bit Position The vector defines the bit position which is complemented to create the data inversion on the write operation. For the creation of 2-bit data inversions, the bit specified by this field plus the odd parity bit of the ECC code are inverted.</p> <p>The RAM controller follows a vector bit ordering scheme where LSB=0. Errors in the ECC syndrome bits can be generated by setting this field to a value greater than the RAM width. For example, consider a 32-bit RAM implementation.</p> <p>The 32-bit ECC approach requires 7 code bits for a 32-bit word. For PRAM data width of 32 bits, the actual SRAM (32b data + 7b for ECC) = 39 bits. The following association between the ERRBIT field and the corrupted memory bit is defined:</p> <p>if ERRBIT = 0, then RAM[0] of the odd bank is inverted if ERRBIT = 1, then RAM[1] of the odd bank is inverted ... if ERRBIT = 31, then RAM[31] of the odd bank is inverted if ERRBIT = 64, then ECC Parity[0] of the odd bank is inverted if ERRBIT = 65, then ECC Parity[1] of the odd bank is inverted ... if ERRBIT = 70, then ECC Parity[6] of the odd bank is inverted</p> <p>For ERRBIT values of 32 to 63 and greater than 70, no bit position is inverted.</p>

If an attempt to force a non-correctable inversion (by asserting EEGR[FRCNCI] or EEGR[FRC1NCI]) and EEGR[ERRBIT] equals 64, then no data inversion will be generated.

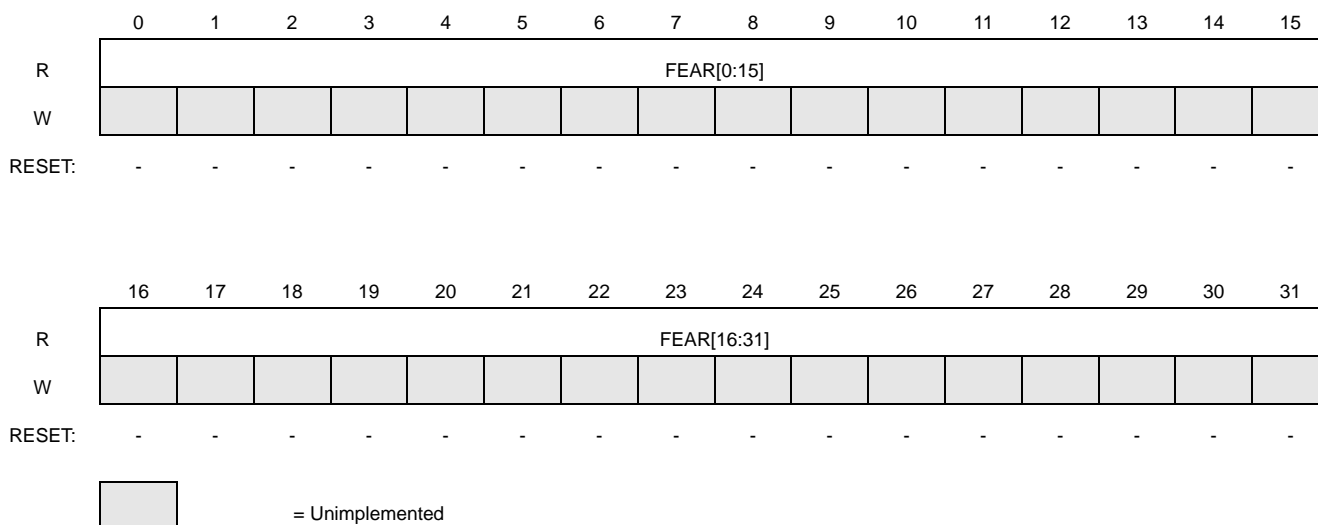
The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior.

16.4.2.10 Flash ECC Address Register (FEAR)

The FEAR is a 32-bit register for capturing the address of the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR, FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 124](#) and [Table 124](#) for the Flash ECC Address Register definition.

Register address: MCM Base + 0x0050


Figure 124. Flash ECC Address (FEAR) Register
Table 124. Flash ECC Address (FEAR) Field Descriptions

Name	Description
0-31 FEAR[0:31]	Flash ECC Address Register This 32-bit register contains the faulting access address of the last, properly-enabled flash ECC event.

16.4.2.11 Flash ECC Master Number Register (FEMR)

The FEMR is a 4-bit register for capturing the AXBS bus master number of the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR, FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 125](#) and [Table 125](#) for the Flash ECC Master Number Register definition.

Register address: MCM Base + 0x0056

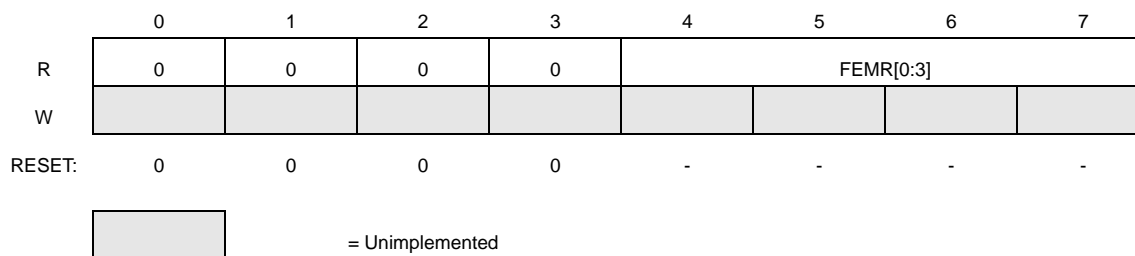

Figure 125. Flash ECC Master Number (FEMR) Register

Table 125. Flash ECC Master Number (FEMR) Field Descriptions

Name	Description
4-7 FEMR[0:3]	Flash ECC Master Number Register This 4-bit register contains the AXBS bus master number of the faulting access of the last, properly-enabled flash ECC event.

16.4.2.12 Flash ECC Attributes (FEAT) register

The FEAT is an 8-bit register for capturing the AXBS bus master attributes of the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR, FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 126](#) and [Table 126](#) for the Flash ECC Attributes Register definition.

Register address: MCM Base + 0x0057

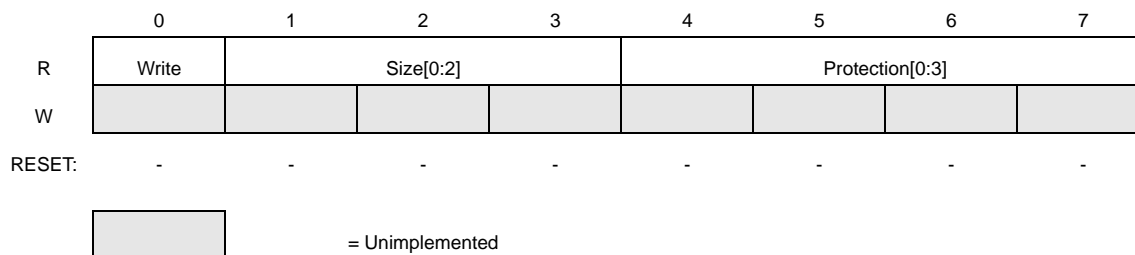


Figure 126. Flash ECC Attributes (FEAT) Register

Table 126. Flash ECC Attributes (FEAT) Field Descriptions

Name	Description
0 Write	AMBA-AHB HWRITE 0 = AMBA-AHB read access 1 = AMBA-AHB write access
1-3 Size[0:2]	AMBA-AHB HSIZE[0:2] 0b000 = 8-bit AMBA-AHB access 0b001 = 16-bit AMBA-AHB access 0b010 = 32-bit AMBA-AHB access 0b1xx = Reserved
4-7 Protection[0:3]	AMBA-AHB HPROT[0:3] Protection[3]: Cacheable 0 = Non-cacheable, 1 = Cacheable Protection[2]: Bufferable 0 = Non-bufferable, 1 = Bufferable Protection[1]: Mode 0 = User mode, 1 = Supervisor mode Protection[0]: Type 0 = I-Fetch, 1 = Data

16.4.2.13 Flash ECC Data Register (FEDR)

The FEDR is a 32-bit register for capturing the data associated with the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR,

FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 127](#) and [Table 127](#) for the Flash ECC Data Register definition.

Register address: MCM Base +0x005C

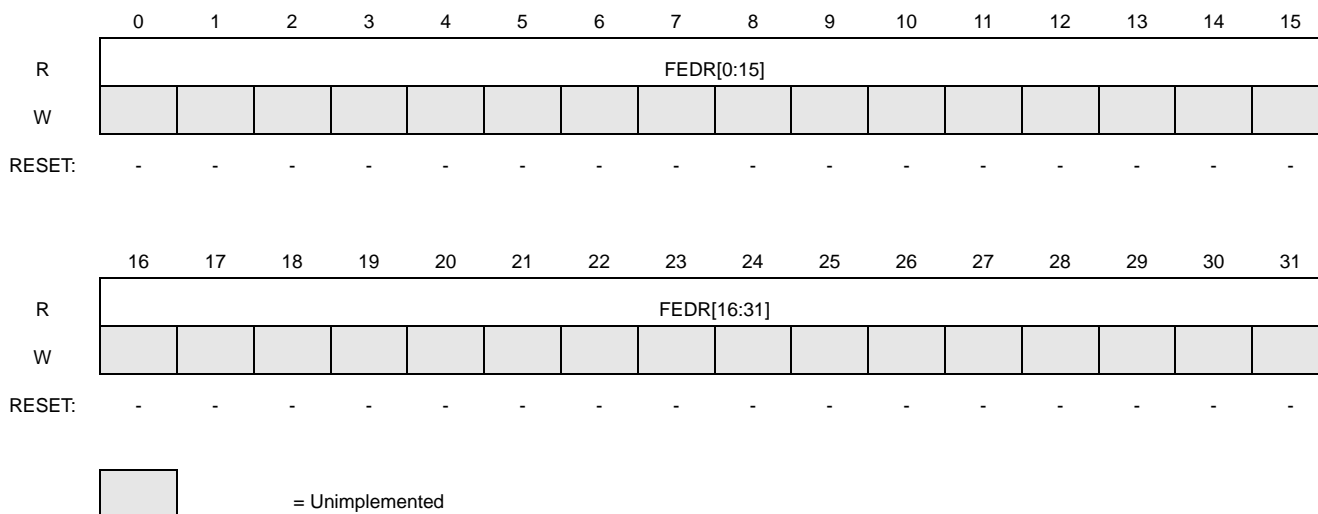


Figure 127. Flash ECC Data (FEDR) Register

Table 127. Flash ECC Data (FEDR) Field Descriptions

Name	Description
0-31 FEDR[0:31]	Flash ECC Data Register This 32-bit register contains the data associated with the faulting access of the last, properly-enabled flash ECC event. The register contains the data value taken directly from the data bus.

16.4.2.14 RAM ECC Address Register (REAR)

The REAR is a 32-bit register for capturing the address of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 128](#) and [Table 128](#) for the RAM ECC Address Register definition.

Register address: MCM Base + 0x0060

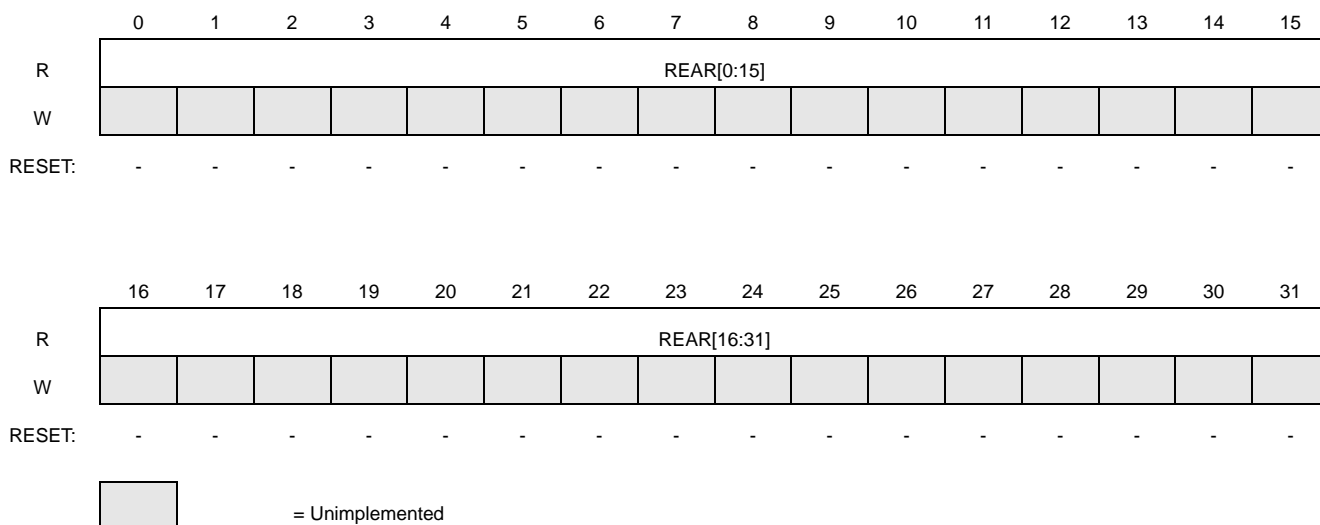


Figure 128. RAM ECC Address (REAR) Register

Table 128. RAM ECC Address (REAR) Field Descriptions

Name	Description
0-31 REAR[0:31]	RAM ECC Address Register This 32-bit register contains the faulting access address of the last, properly-enabled RAM ECC event.

16.4.2.15 RAM ECC Syndrome Register (RESR)

The RESR is an 8-bit register for capturing the error syndrome of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 129](#) and [Table 129](#) for the RAM ECC Syndrome Register definition.

Register address: MCM Base + 0x0065



Figure 129. RAM ECC Syndrome (RESR) Register

Table 129. RAM ECC Syndrome (RESR) Field Descriptions

Name	Description
0-7 RESR[0:7]	<p>RAM ECC Syndrome Register</p> <p>This 8-bit syndrome field includes 6 bits of Hamming decoded parity plus an odd-parity bit for the entire 39-bit (32-bit data + 7 ECC) code word. The upper 7 bits of the syndrome specify the exact bit position in error for single-bit correctable codewords, and the combination of a non-zero 7-bit syndrome plus overall incorrect parity bit signal a multi-bit, non-correctable error.</p> <p>For correctable single-bit errors, the mapping shown in Table 129 associates the upper 7 bits of the syndrome with the data bit in error.</p>

Note: [Table 129](#) associates the upper 7 bits of the ECC syndrome with the exact data bit in error for single-bit correctable codewords. This table follows the bit vectoring notation where the LSB=0. Note that the syndrome value of 0x0001 implies no error condition but this value is not readable when the PRESR is read for the no error case.

Table 130. RAM Syndrome Mapping for Single-Bit Correctable Errors

RESR[0:7]	Data Bit in Error
0x0000	ECC ODD[0]
0x0001	No Error
0x0002	ECC ODD[1]
0x0004	ECC ODD[2]
0x0006	DATA ODD BANK[31]
0x0008	ECC ODD[3]
0x000a	DATA ODD BANK[30]
0x000c	DATA ODD BANK[29]
0x000e	DATA ODD BANK[28]
0x0010	ECC ODD[4]
0x0012	DATA ODD BANK[27]
0x0014	DATA ODD BANK[26]
0x0016	DATA ODD BANK[25]
0x0018	DATA ODD BANK[24]
0x001a	DATA ODD BANK[23]
0x001c	DATA ODD BANK[22]
0x0050	DATA ODD BANK[21]
0x0020	ECC ODD[5]
0x0022	DATA ODD BANK[20]
0x0024	DATA ODD BANK[19]
0x0026	DATA ODD BANK[18]
0x0028	DATA ODD BANK[17]
0x002a	DATA ODD BANK[16]
0x002c	DATA ODD BANK[15]
0x0058	DATA ODD BANK[14]
0x0030	DATA ODD BANK[13]
0x0032	DATA ODD BANK[12]
0x0034	DATA ODD BANK[11]
0x0064	DATA ODD BANK[10]

Table 130. RAM Syndrome Mapping for Single-Bit Correctable Errors (continued)

RESR[0:7]	Data Bit in Error
0x0038	DATA ODD BANK[9]
0x0062	DATA ODD BANK[8]
0x0070	DATA ODD BANK[7]
0x0060	DATA ODD BANK[6]
0x0040	ECC ODD[6]
0x0042	DATA ODD BANK[5]
0x0044	DATA ODD BANK[4]
0x0046	DATA ODD BANK[3]
0x0048	DATA ODD BANK[2]
0x004a	DATA ODD BANK[1]
0x004c	DATA ODD BANK[0]
0x0003,0x0005.....0x004d	Multiple bit error
> 0x004d	Multiple bit error

16.4.2.16 RAM ECC Master Number Register (REMR)

The REMR is a 4-bit register for capturing the AXBS bus master number of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (RIBC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 130](#) and [Table 131](#) for the RAM ECC Master Number Register definition.

Register address: MCM Base + 0x0066

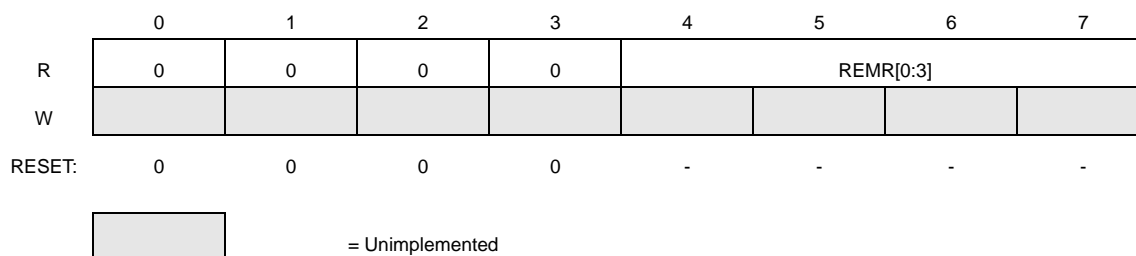


Figure 130. RAM ECC Master Number (REMR) Register

Table 131. RAM ECC Master Number (REMR) Field Descriptions

Name	Description
4-7 REMR[0:3]	RAM ECC Master Number Register This 4-bit register contains the AXBS bus master number of the faulting access of the last, properly-enabled RAM ECC event.

16.4.2.17 RAM ECC Attributes (REAT) register

The REAT is an 8-bit register for capturing the AXBS bus master attributes of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 131](#) and [Table 132](#) for the RAM ECC Attributes Register definition.

Register address: MCM Base + 0x0067

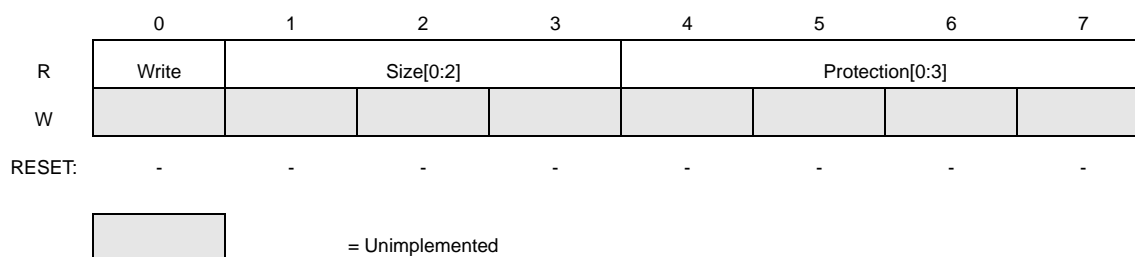


Figure 131. RAM ECC Attributes (REAT) Register

Table 132. RAM ECC Attributes (REAT) Field Descriptions

Name	Description
0 Write	AMBA-AHB HWRITE 0 = AMBA-AHB read access 1 = AMBA-AHB write access
1-3 Size[0:2]	AMBA-AHB HSIZE[0:2] 0b000 = 8-bit AMBA-AHB access 0b001 = 16-bit AMBA-AHB access 0b010 = 32-bit AMBA-AHB access 0b1xx = Reserved
4-7 Protection[0:3]	AMBA-AHB HPROT[0:3] Protection[3]: Cacheable 0 = Non-cacheable, 1 = Cacheable Protection[2]: Bufferable 0 = Non-bufferable, 1 = Bufferable Protection[1]: Mode 0 = User mode, 1 = Supervisor mode Protection[0]: Type 0 = I-Fetch, 1 = Data

16.4.2.18 RAM ECC Data Register (REDR)

The REDR is a 32-bit register for capturing the data associated with the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

This register can only be read from the IPS programming model; any attempted write is ignored. See [Figure 132](#) and [Table 133](#) for the RAM ECC Data Register definition.

Miscellaneous Control Module (MCM)

Register address: MCM Base +0x006c

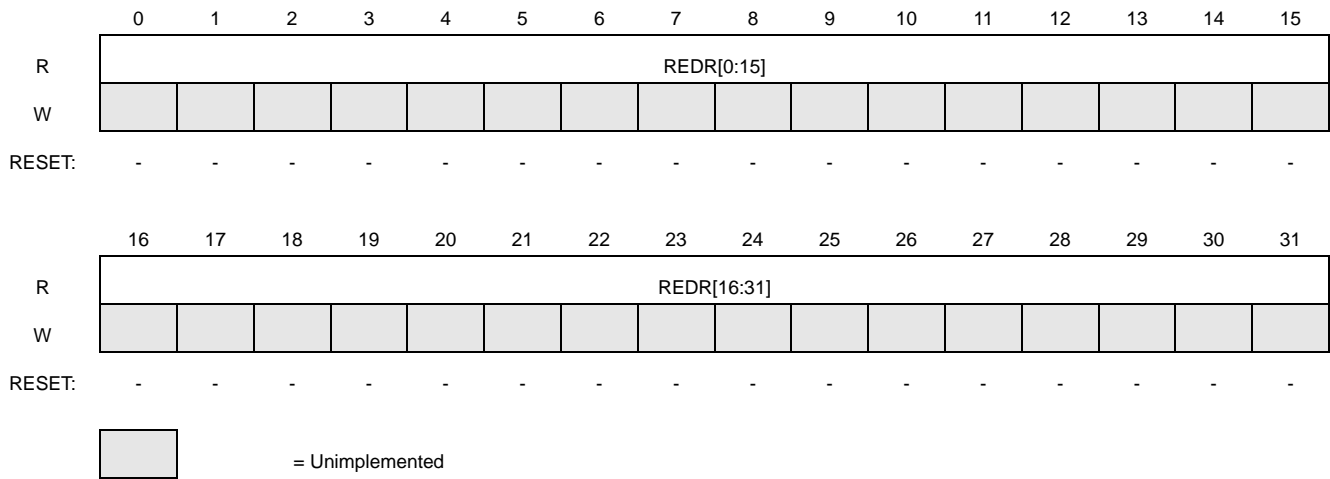


Figure 132. RAM ECC Data (REDR) Register

Table 133. RAM ECC Data (REDR) Field Descriptions

Name	Description
0-31 REDR[0:31]	RAM ECC Data Register This 32-bit register contains the data associated with the faulting access of the last, properly-enabled RAM ECC event. The register contains the data value taken directly from the data bus.

16.4.3 MCM_reg_protection

The MCM_reg_protection logic provides hardware enforcement of supervisor mode access protection for four on-platform IPS modules: INTC, MCM, STM, and SWT. This logic resides between the on-platform bus sourced by the AIPS bus controller and the individual slave modules. It monitors the bus access type (supervisor or user) and if a user access is attempted, the transfer is terminated with an error and inhibited from reaching the slave module. Identical logic is replicated for each of the five, targeted slave modules. A block diagram of the MCM_reg_protection module is shown in [Figure 133](#).

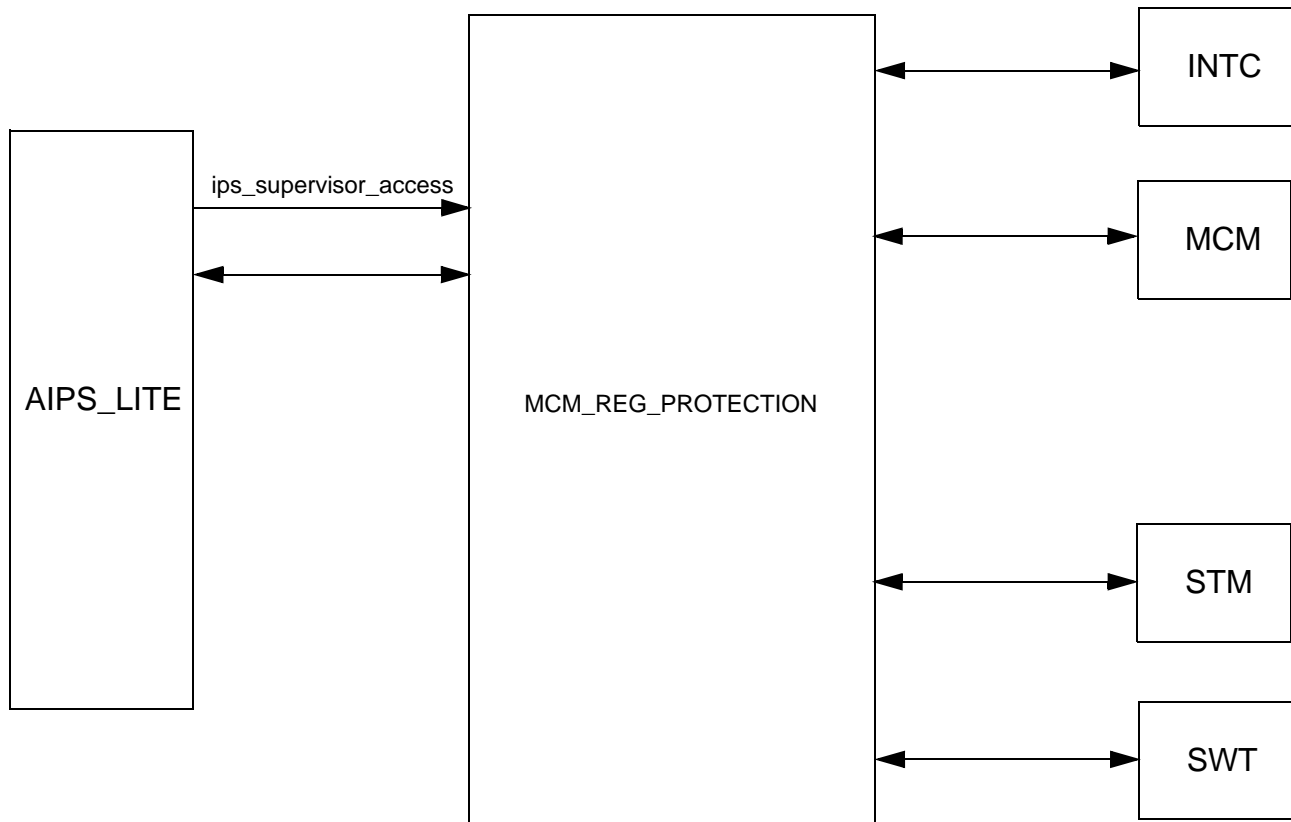


Figure 133. Spp_Ips_Reg_Protection block diagram

Attempted accesses to reserved addresses result in an error termination, while attempted writes to read-only registers are ignored and do not terminate with an error. Unless noted otherwise, writes to the programming model must match the size of the register; for example, an n -bit register only supports n -bit writes, etc. Attempted writes of a different size than the register width produce an error termination of the bus cycle and no change to the targeted register.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 17

Internal Static RAM (SRAM)

17.1 Introduction

The general-purpose SRAM has a size of 96 KB.

The SRAM provides the following features:

- SRAM can be read/written from any bus master
- Byte, halfword, word and doubleword addressable
- Single-bit correction and double-bit error detection

17.2 SRAM operating mode

The SRAM has only one operating mode. No standby mode is available.

Table 134. SRAM operating modes

Mode	Configuration
Normal (functional)	Allows reads and writes of SRAM

17.3 Register memory map

The SRAM occupies 96 KB of memory starting at the base address as shown in [Table 135](#).

Table 135. SRAM memory map

Address	Register name	Register description	Size
0x4000_0000 (Base)	—	—	96 KB

The internal SRAM has no registers. Registers for the SRAM ECC are located in the MCM .

17.4 SRAM ECC mechanism

The SRAM ECC detects the following conditions and produces the following results:

- Detects and corrects all 1-bit errors
- Detects and flags all 2-bit errors as non-correctable errors
- Detects 39-bit reads (32-bit data bus plus the 7-bit ECC) that return all zeros or all ones, asserts an error indicator on the bus cycle, and sets the error flag

SRAM does not detect all errors greater than 2 bits.

Internal SRAM write operations are performed on the following byte boundaries:

- 1 byte (0:7 bits)
- 2 bytes (0:15 bits)
- 4 bytes or 1 word (0:31 bits)

If the entire 32 data bits are written to SRAM, no read operation is performed and the ECC is calculated across the 32-bit data bus. The 8-bit ECC is appended to the data segment and written to SRAM.

If the write operation is less than the entire 32-bit data width (1 or 2-byte segment), the following occurs:

1. The ECC mechanism checks the entire 32-bit data bus for errors, detecting and either correcting or flagging errors.
2. The write data bytes (1 or 2-byte segment) are merged with the corrected 32 bits on the data bus.
3. The ECC is then calculated on the resulting 32 bits formed in the previous step.
4. The 7-bit ECC result is appended to the 32 bits from the data bus, and the 39-bit value is then written to SRAM.

17.4.1 Access timing

The system bus is a two-stage pipelined bus that makes the timing of any access dependent on the access during the previous clock. [Table 136](#) lists the various combinations of read and write operations to SRAM and the number of wait states used for the each operation. The table columns contain the following information:

- Current operation—Lists the type of SRAM operation currently executing
- Previous operation—Lists the valid types of SRAM operations that can precede the current SRAM operation (valid operation during the preceding clock)
- Wait states—Lists the number of wait states (bus clocks) the operation requires, which depends on the combination of the current and previous operation

Table 136. Number of wait states required for SRAM operations

Operation type	Current operation	Previous operation	Number of wait states required
Read	Read	Idle	1
		Pipelined read	
		8 , 16 or 32-bit write	0 (read from the same address)
			1 (read from a different address)
	Pipelined read	Read	0

Table 136. Number of wait states required for SRAM operations (continued)

Operation type	Current operation	Previous operation	Number of wait states required
Write	8 or 16-bit write	Idle	1
		Read	
		Pipelined 8- or 16-bit write	2
	32-bit write		
	8 or 16-bit write	0 (write to the same address)	
	Pipelined 8, 16 or 32-bit write	8 , 16 or 32-bit write	0
	32-bit write	Idle	0
32-bit write			
Read			

17.4.2 Reset effects on SRAM accesses

Asynchronous reset will possibly corrupt RAM if it asserts during a read or write operation to SRAM. The completion of that access depends on the cycle at which the reset occurs. If no access is occurring when reset occurs, RAM corruption does not happen.

Instead synchronous reset (SW reset) should be used in controlled function (without RAM accesses) in case initialization procedure is needed without RAM initialization.

17.5 Functional description

ECC checks are performed during the read portion of an SRAM ECC read/write (R/W) operation, and ECC calculations are performed during the write portion of a R/W operation. Because the ECC bits can contain random data after the device is powered on, the SRAM must be initialized by executing 32-bit write operations prior any read accesses. This is also true for implicit read accesses caused by any write accesses smaller than 32 bits as discussed in [Section 17.4, “SRAM ECC mechanism”](#).

17.6 Initialization and application information

To use the SRAM, the ECC must check all bits that require initialization after power on. All writes must specify an even number of registers performed on 32-bit word-aligned boundaries. If the write is not the entire 32-bits (8 or 16 bits), a read/modify/write operation is generated that checks the ECC value upon the read. Refer to [Section 17.4, “SRAM ECC mechanism”](#).

NOTE

You *must* initialize SRAM, even if the application does not use ECC reporting.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 18

Flash Memory

18.1 Introduction

The flash memory comprises a platform flash controller interface and two flash memory arrays: one array of 512 KB for code (code flash) and one array of 64 KB for data (data flash). The flash architecture of the MPC5604E device is illustrated in [Figure 134](#).

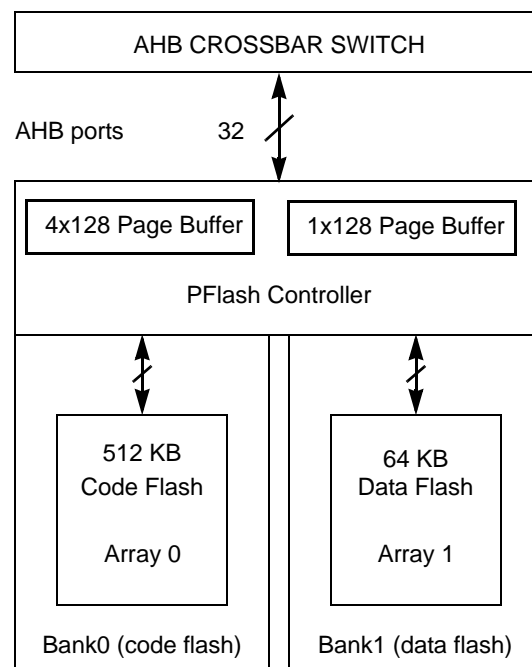


Figure 134. MPC5604E flash memory architecture

MPC5604E flash memory is arranged as follows:

Array0 (code flash):

- 512 KB + 16 KB shadow block + 16 KB test block
- 8 small blocks organized as 16 KB, 16 KB, 32 KB, 32 KB, 16 KB, 16 KB, 64 KB and 64 KB
- 2 large blocks organized as 128 KB and 128 KB
- 1 Shadow block, 16 KB
- 1 Test block, 16 KB

Array1 (data flash):

- 64 KB + 8 KB test block

- 4 small blocks organized as 16 KB, 16 KB, 16 KB, 16 KB,
- 1 Test block, 8 KB

18.2 Platform flash controller

18.2.1 Introduction

This section provides an introduction of the platform flash controller, which acts as the interface between the system bus and as many as two banks of flash memory arrays (program and data). It intelligently converts the protocols between the system bus and the dedicated flash array interfaces. Several important terms are used to describe the platform flash controller module and its connections. These terms are defined here.

- **Port**—This term describes the AMBA-AHB connection(s) into the platform flash controller. From an architectural and programming model viewpoint, the definition supports as many as two AHB ports, even though this specific controller only supports a single AHB connection.
- **Bank**—This term describes the attached flash memories. From the platform flash controller's perspective, there may be one or two attached banks of flash memory. The code flash bank is required and always attached to bank0. Additionally, there is a data flash attached to bank1. The platform flash controller interface supports two separate connections, one to each memory bank. On the MPC5604E device, bank0 and bank1 are internal to the device.
- **Array**—Each memory bank has one flash array instantiation.
- **Page**—This value defines the number of bits read from the flash array in a single access. For this controller and memory, the page size is 128 bits (16 bytes).

The nomenclature “page buffers” and “line buffers” are used interchangeably.

18.2.1.1 Overview

The platform flash controller supports a 32-bit data bus width at the AHB port and connections to 128-bit read data interfaces from two memory banks, where each bank contains one instantiation of the flash memory array. One flash bank is connected to the code flash memory and the other bank is connected to the data flash memory. The memory controller capabilities vary between the two banks with each bank's functionality optimized with the typical use cases associated with the attached flash memory. As an example, the platform flash controller logic associated with the code flash bank contains a four-entry “page” buffer, each entry containing 128 bits of data (1 flash page) plus an associated controller that prefetches sequential lines of data from the flash array into the buffer, while the controller logic associated with the data flash bank only supports a 128-bit register that serves as a temporary page holding register and does not support any prefetching. Prefetch buffer hits from the code flash bank support 0-wait AHB data phase responses. AHB read requests that miss the buffers generate the needed flash array access and are forwarded to the AHB upon completion, typically incurring two wait states at an operating frequency of 60 to 64 MHz.

This memory controller is optimized for applications where a cacheless processor core, for example the Power e200z0h, is connected through the platform to on-chip memories, for example flash and RAM, where the processor and platform operate at the same frequency. For these applications, the 2-stage

pipeline AMBA-AHB system bus is effectively mapped directly into stages of the processor's pipeline and 0 wait state responses for most memory accesses are critical for providing the required level of system performance.

18.2.1.2 Features

The following list summarizes the key features of the platform flash controller:

- Single AHB port interface supports a 32-bit data bus. All AHB aligned and unaligned reads within the 32-bit container are supported. Only aligned word writes are supported.
- Array interfaces support a 128-bit read data bus and a 64-bit write data bus for each bank.
- Interface with code flash provides configurable read buffering and page prefetch support. Four page read buffers (each 128 bits wide) and a prefetch controller support single-cycle read responses (0 AHB data phase wait states) for hits in the buffers. The buffers implement a least-recently-used replacement algorithm to maximize performance.
- Interface with data flash includes a 128-bit register to temporarily hold a single flash page. This logic supports single-cycle read responses (0 AHB data phase wait states) for accesses that hit in the holding register. There is no support for prefetching associated with bank 1.
- Programmable response for read-while-write sequences including support for stall-while-write, optional stall notification interrupt, optional flash operation termination, and optional termination notification interrupt
- Separate and independent configurable access timing (on a per bank basis) to support use across a wide range of platforms and frequencies
- Support of address-based read access timing for emulation of other memory types
- Support for reporting of single- and multi-bit flash ECC events
- Typical operating configuration loaded into programming model by system reset

18.2.2 Modes of operation

The platform flash controller module does not support any special modes of operation. Its operation is driven from the AMBA-AHB memory references it receives from the platform's bus masters. Its configuration is defined by the setting of the programming model registers, physically located as part of the flash array modules.

18.2.3 External signal descriptions

The platform flash controller does not directly interface with any external signals. Its primary internal interfaces include a connection to an AMBA-AHB crossbar (or memory protection unit) slave port and connections with as many as two banks (code and data) of flash memory, each containing one instantiation of the flash array. Additionally, the operating configuration for the platform flash controller is defined by the contents of certain code flash array0 registers that are inputs to the module.

18.2.4 Memory map and registers description

Two memory maps are associated with the platform flash controller: one for the flash memory space and another for the program-visible control and configuration registers. The flash memory space is accessed via the AMBA-AHB port. The program-visible registers are accessed via the slave peripheral bus. Details on both memory spaces are provided in [Section 18.2.4.1, “Memory map”](#).

There are no program-visible registers that physically reside inside the platform flash controller. Rather, the platform flash controller receives control and configuration information from the flash array controller(s) to determine the operating configuration. These are part of the flash array’s configuration registers mapped into its slave peripheral (IPS) address space but are described here.

18.2.4.1 Memory map

First, consider the flash memory space accessed via transactions from the platform flash controller’s AHB port. To support the two separate flash memory banks, the platform flash controller uses address bit 23 (haddr[23]) to steer the access to the appropriate memory bank. In addition to the actual flash memory regions, there are shadow and test sectors included in the system memory map. The program-visible control and configuration registers associated with each memory array are included in the slave peripheral address region. The system memory map defines one code flash array and one data flash array. See [Table 137](#).

Table 137. Flash-related regions in the system memory map

Start address	End address	Size (KB)	Region
0x0000_0000	0x0000_3FFF	16	Code Flash Array 0
0x0000_4000	0x0000_7FFF	16	
0x0000_8000	0x0000_FFFF	32	
0x0001_0000	0x0000_17FF	32	
0x0001_8000	0x0001_BFFF	16	
0x0001_C000	0x0001_FFFF	16	
0x0002_0000	0x0002_FFFF	64	
0x0003_0000	0x0003_FFFF	64	
0x0004_0000	0x0005_FFFF	128	
0x0006_0000	0x0007_FFFF	128	
0x0008_0000	0x001F_FFFF	1536	
0x0020_0000	0x0020_3FFF	16	Code Flash Array 0 Shadow Sector
0x0020_4000	0x003F_FFFF	2032	Reserved
0x0040_0000	0x0040_3FFF	16	Code Flash Array 0 Test Sector
0x0040_4000	0x005F_FFFF	2032	Reserved
0x0080_0000	0x0080_3FFF	16	Data Flash Array 0

Table 137. Flash-related regions in the system memory map (continued)

Start address	End address	Size (KB)	Region
0x0080_4000	0x0080_7FFF	16	Data Flash Array 0
0x0080_8000	0x0080_BFFF	16	Data Flash Array 0
0x0080_C000	0x0080_FFFF	16	Data Flash Array 0
0x0081_0000	0x009F_FFFF	1984	Reserved
0x00A0_0000	0x00BF_FFFF	2048	Reserved
0x00C0_0000	0x00C0_1FFF	8	Reserved
0x00C0_2000	0x00C0_3FFF	8	Data Flash Test Sector
0x00C0_4000	0x00FF_FFFF	4080	Reserved

For additional information on the address-based read access timing for emulation of other memory types, see [Section 18.2.17, “Wait state emulation”](#).

Next, consider the memory map associated with the control and configuration registers.

There are registers that control operation of the platform flash controller. Note the first two flash array registers (PFCR0, PFCR1) are reset to a device-defined value, while the remaining register (PFAPR) is loaded at reset from specific locations in the array’s shadow region.

Regardless of the number of populated banks or the number of flash arrays included in a given bank, the configuration of the platform flash controller is wholly specified by the platform flash controller control registers associated with code flash array0. The code array0 register settings define the operating behavior of **both** flash banks. It is recommended to set the platform flash controller control registers for both arrays to the array0 values.

NOTE

To perform program and erase operations, the control registers in the actual referenced flash array must both be programmed, but the configuration of the platform flash controller module is defined by the platform flash controller control registers of code array0.

The 32-bit memory map for the platform flash controller control registers is shown in [Table 138](#).

Table 138. Platform Flash Controller 32-bit memory map

Offset from PFLASH_BASE (0xFFE8_8000)	Register	Access
0x001C	Platform Flash Configuration Register 0 (PFCR0)	R/W
0x0020	Platform Flash Configuration Register 1 (PFCR1)	R/W
0x0024	Platform Flash Access Protection Register (PFAPR)	R/W

18.2.4.2 Registers description

This section details the individual registers of the platform flash controller. The platform flash registers control flash behavior globally.

18.2.4.2.1 Platform Flash Configuration Register 0 (PFCR0)

The Platform Flash Configuration Register 0 (PFCR0) defines the configuration associated with flash memory bank0, which corresponds to the code flash. The register is described in [Figure 135](#) and [Table 139](#).

NOTE

This register is not implemented on the data flash block.

PFCR0[BK0_APC] must equal to PFCR0[BK0_RWSC]. Refer datasheet for correct setting of RWSC.

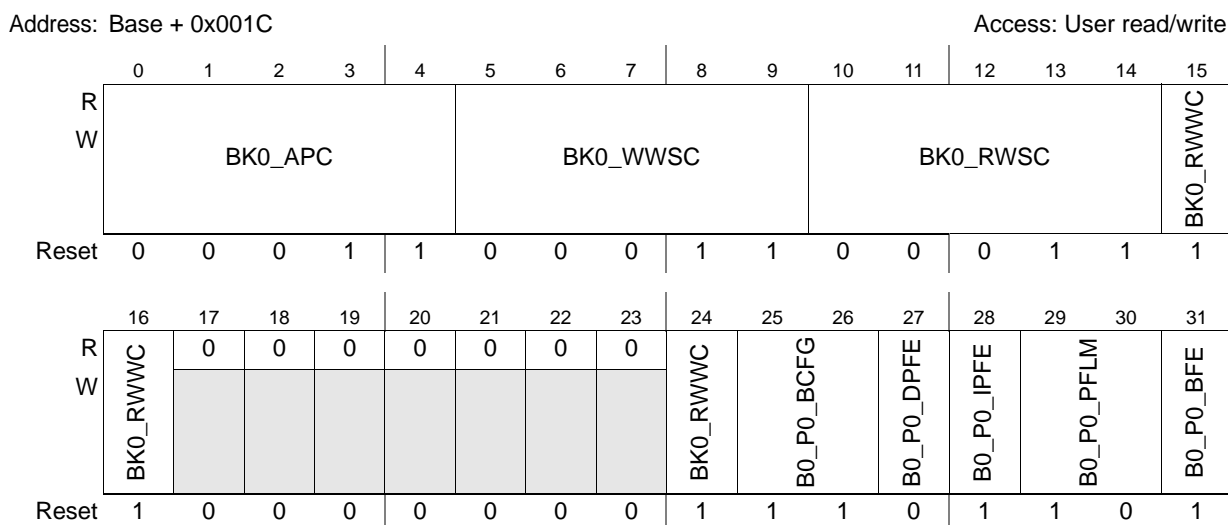


Figure 135. Platform Flash Configuration Register 0 (PFCR0)

Table 139. PFCR0 field descriptions

Field	Description
BK0_APC	<p>Bank0 Address Pipelining Control</p> <p>This field controls the number of cycles between flash array access requests. This field must be set to a value appropriate to the operating frequency of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 0b00010 by hardware reset.</p> <p>00000 Accesses may be initiated on consecutive (back-to-back) cycles. 00001 Access requests require one additional hold cycle. 00010 Access requests require two additional hold cycles. ... 11110 Access requests require 30 additional hold cycles. 11111 Access requests require 31 additional hold cycles.</p>

Table 139. PFCR0 field descriptions (continued)

Field	Description
BK0_WWSC	<p>Bank0 Write Wait State Control This field controls the number of wait states to be added to the flash array access time for writes. This field must be set to a value appropriate to the operating frequency of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to an appropriate value by hardware reset. This field is set to 0b00010 by hardware reset.</p> <p>00000 No additional wait states are added. 00001 1 additional wait state is added. 00010 2 additional wait states are added. ... 111111 31 additional wait states are added.</p>
BK0_RWSC	<p>Bank0 Read Wait State Control This field controls the number of wait states to be added to the flash array access time for reads. This field must be set to a value corresponding to the operating frequency of the PFLASH and the actual read access time of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation.</p> <p>0 MHz, < 23 MHz APC = RWSC = 0. 23 MHz, < 45 MHz APC = RWSC = 1. 45 MHz, < 68 MHz APC = RWSC = 2. 68 MHz, < 90 MHz APC = RWSC = 3.</p> <p>This field is set to 0b00010 by hardware reset.</p> <p>00000 No additional wait states are added. 00001 1 additional wait state is added. 00010 2 additional wait states are added. ... 111111 31 additional wait states are added.</p>
BK0_RWWC	<p>Bank0 Read-While-Write Control This 3-bit field defines the controller response to flash reads while the array is busy with a program (write) or erase operation.</p> <p>0xx Reserved. This configuration should be avoided. 100 Generate a bus stall for a read while write/erase, enable the operation termination and the abort notification interrupt. 101 Generate a bus stall for a read while write/erase, enable the operation abort, disable the abort notification interrupt. 110 Generate a bus stall for a read while write/erase, enable the stall notification interrupt, disable the abort + abort notification interrupt. 111 Generate a bus stall for a read while write/erase, disable the stall notification interrupt, disable the abort + abort notification interrupt.</p> <p>This field is set to 0b111 by hardware reset enabling the stall-while-write/erase and disabling the abort and notification interrupts.</p>
	Reserved

Table 139. PFCR0 field descriptions (continued)

Field	Description
B0_P0_BCFG	<p>Bank0, Port 0 Page Buffer Configuration</p> <p>This field controls the configuration of the four page buffers in the PFLASH controller. The buffers can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers.</p> <p>If enabled, when a buffer miss occurs, it is allocated to the least-recently-used buffer within the group and the just-fetched entry then marked as most-recently-used. If the flash access is for the next-sequential line, the buffer is not marked as most-recently-used until the given address produces a buffer hit.</p> <p>00 All four buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved. 10 The buffers are partitioned into two groups with buffers 0 and 1 allocated for instruction fetches and buffers 2 and 3 for data accesses. 11 The buffers are partitioned into two groups with buffers 0,1,2 allocated for instruction fetches and buffer 3 for data accesses.</p> <p>This field is set to 2b11 by hardware reset.</p>
B0_P0_DPFE	<p>Bank0, Port 0 Data Prefetch Enable</p> <p>This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset.</p> <p>0 No prefetching is triggered by a data read access. 1 If page buffers are enabled (B0_P0_BFE = 1), prefetching is triggered by any data read access.</p>
B0_P0_IPFE	<p>Bank0, Port 0 Instruction Prefetch Enable</p> <p>This field enables or disables prefetching initiated by an instruction fetch read access. This field is set by hardware reset.</p> <p>0 No prefetching is triggered by an instruction fetch read access. 1 If page buffers are enabled (B0_P0_BFE = 1), prefetching is triggered by any instruction fetch read access.</p>
B0_P0_PFLM	<p>Bank0, Port 0 Prefetch Limit</p> <p>This field controls the prefetch algorithm used by the PFLASH controller. This field defines the prefetch behavior. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is set to 2b10 by hardware reset.</p> <p>00 No prefetching is performed. 01 The referenced line is prefetched on a buffer miss, that is, <i>prefetch on miss</i>. 1x The referenced line is prefetched on a buffer miss, or the next sequential page is prefetched on a buffer hit (if not already present), that is, <i>prefetch on miss or hit</i>.</p>
B0_P0_BFE	<p>Bank0, Port 0 Buffer Enable</p> <p>This bit enables or disables page buffer read hits. It is also used to invalidate the buffers. This bit is set by hardware reset.</p> <p>0 The page buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The page buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.</p>

18.2.4.2.2 Platform Flash Configuration Register 1 (PFCR1)

The Platform Flash Configuration Register 1 (PFCR1) defines the configuration associated with flash memory bank1. This corresponds to the data flash. The register is described in [Figure 136](#) and [Table 140](#).

NOTE

This register is not implemented on the data flash block.

PFCR1[BK1_APC] must equal to PFCR1[BK1_RWSC]. Refer datasheet for correct setting of RWSC.

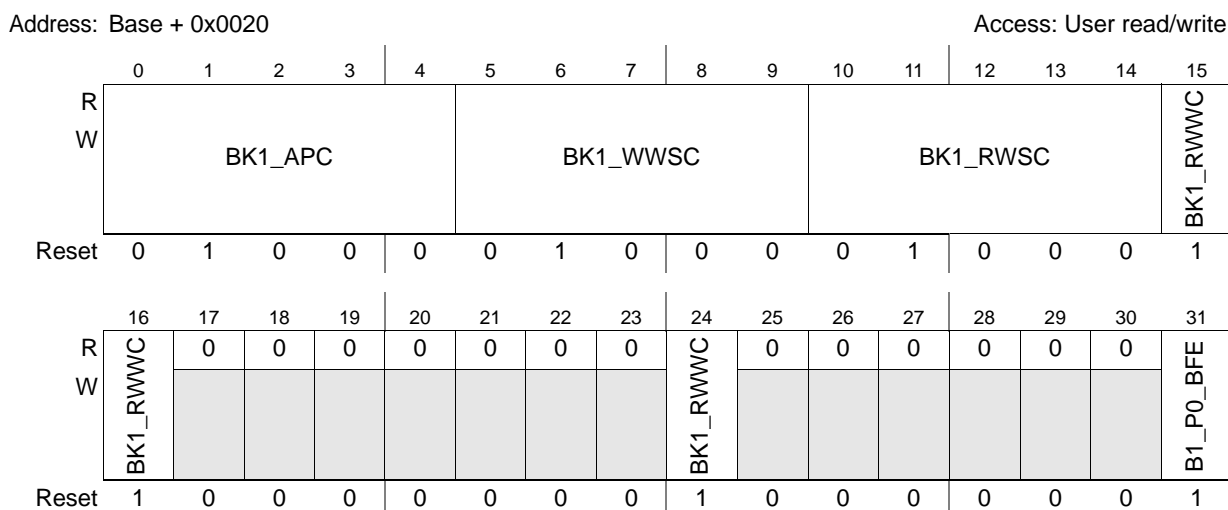


Figure 136. Platform Flash Configuration Register 1 (PFCR1)

Table 140. PFCR1 field descriptions

Field	Description
BK1_APC	<p>Bank1 Address Pipelining Control</p> <p>This field controls the number of cycles between flash array access requests. This field must be set to a value appropriate to the operating frequency of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 0b00010 by hardware reset.</p> <p>00000 Accesses may be initiated on consecutive (back-to-back) cycles. 00001 Access requests require one additional hold cycle. 00010 Access requests require two additional hold cycles. ... 11110 Access requests require 30 additional hold cycles. 11111 Access requests require 31 additional hold cycles.</p>

Table 140. PFCR1 field descriptions (continued)

Field	Description
BK1_WWSC	<p>Bank1 Write Wait State Control This field controls the number of wait states to be added to the flash array access time for writes. This field must be set to a value appropriate to the operating frequency of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to an appropriate value by hardware reset. This field is set to 0b00010 by hardware reset.</p> <p>00000 No additional wait states are added. 00001 1 additional wait state is added. 00010 2 additional wait states are added. ... 111111 31 additional wait states are added.</p>
BK1_RWSC	<p>Bank1 Read Wait State Control This field controls the number of wait states to be added to the flash array access time for reads. This field must be set to a value corresponding to the operating frequency of the PFLASH and the actual read access time of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation.</p> <p>0 MHz, < 23 MHz APC = RWSC = 0. 23 MHz, < 45 MHz APC = RWSC = 1. 45 MHz, < 68 MHz APC = RWSC = 2. 68 MHz, < 90 MHz APC = RWSC = 3.</p> <p>This field is set to 0b00010 by hardware reset.</p> <p>00000 No additional wait states are added. 00001 1 additional wait state is added. 00010 2 additional wait states are added. ... 111111 31 additional wait states are added.</p>
BK1_RWWC	<p>Bank1 Read-While-Write Control This 3-bit field defines the controller response to flash reads while the array is busy with a program (write) or erase operation.</p> <p>0xx Reserved. This configuration should be avoided. 100 Generate a bus stall for a read while write/erase, enable the operation abort and the abort notification interrupt. 101 Generate a bus stall for a read while write/erase, enable the operation abort, disable the abort notification interrupt. 110 Generate a bus stall for a read while write/erase, enable the stall notification interrupt, disable the abort + abort notification interrupt. 111 Generate a bus stall for a read while write/erase, disable the stall notification interrupt, disable the abort + abort notification interrupt.</p> <p>This field is set to 0b111 by hardware reset enabling the stall-while-write/erase and disabling the abort and notification interrupts.</p>
	Reserved, should be cleared.

Table 140. PFCR1 field descriptions (continued)

Field	Description
B1_P0_PFE	Bank1, Port 0 Buffer Enable This bit enables or disables read hits from the 32-bit holding register. It is also used to invalidate the contents of the holding register. This bit is set by hardware reset, enabling the use of the holding register. 0 The holding register is disabled from satisfying read requests. 1 The holding register is enabled to satisfy read requests on hits.

18.2.4.2.3 Platform Flash Access Protection Register (PFAPR)

The Platform Flash Access Protection Register (PFAPR) controls read and write accesses to the flash based on system master number. Prefetching capabilities are defined on a per master basis. This register also defines the arbitration mode for controllers supporting two AHB ports. The register is described in [Figure 137](#) and [Table 141](#).

The contents of the register are loaded from location 0x20_3E00 of the shadow region in the code flash (bank0) array at reset. To temporarily change the values of any of the fields in the PFAPR, a write to the IPS-mapped register is performed. To change the values loaded into the PFAPR *at reset*, the word location at address 0x20_3E00 of the shadow region in the flash array must be programmed using the normal sequence of operations. The reset value shown in [Table 137](#) reflects an erased or unprogrammed value from the shadow region.

NOTE

This register is not implemented on the data flash block.

Address: Base + 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	ARBM		0	0	0	M4	M3	M2	M1	M0
W												PFD	PFD	PFD	PFD	PFD
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M7AP		M6AP		M5AP		M4AP		M3AP		M2AP		M1AP		M0AP	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 137. Platform Flash Access Protection Register (PFAPR)
Table 141. PFAPR field descriptions

Field	Description
	Reserved, should be cleared.

Table 141. PFAPR field descriptions (continued)

Field	Description
ARBM	<p>Arbitration Mode This 2-bit field controls the arbitration for PFLASH controllers supporting 2 AHB ports.</p> <p>00 Fixed priority arbitration with AHB p0 > p1. 01 Fixed priority arbitration with AHB p1 > p0. 1x Round-robin arbitration.</p>
MxPFD	<p>Master x Prefetch Disable ($x = 0,1,2,\dots,7$) These bits control whether prefetching may be triggered based on the master number of the requesting AHB master. This field is further qualified by the PFCR0[B0_Px_DPFE, B0_Px_IPFE, Bx_Py_BFE] bits.</p> <p>0 Prefetching may be triggered by this master. 1 No prefetching may be triggered by this master.</p>
MxAP	<p>Master x Access Protection ($x = 0,1,2,\dots,7$) These fields control whether read and write accesses to the flash are allowed based on the master number of the initiating module.</p> <p>00 No accesses may be performed by this master. 01 Only read accesses may be performed by this master. 10 Only write accesses may be performed by this master. 11 Both read and write accesses may be performed by this master.</p>

18.2.5 Functional description

The platform flash controller interfaces between the AHB-Lite 2.v6 system bus and the flash memory arrays.

The platform flash controller generates read and write enables, the flash array address, write size, and write data as inputs to the flash array. The platform flash controller captures read data from the flash array interface and drives it onto the AHB. As much as four pages of data (128-bit width) from bank0 are buffered by the platform flash controller. Lines may be prefetched in advance of being requested by the AHB interface, allowing single-cycle (0 AHB wait states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling page read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

Throughout this discussion, bn_n is used as a prefix to refer to two signals, each for each bank: $bk0_n$ and $bk1_n$. Also, the nomenclature $Bx_Py_RegName$ is used to reference a program-visible register field associated with bank “x” and port “y”.

18.2.6 Basic interface protocol

The platform flash controller interfaces to the flash array by driving addresses ($bkn_fl_addr[23:0]$) and read or write enable signals ($bkn_fl_rd_en$, $bkn_fl_wr_en$).

The read or write enable signal ($bkn_fl_rd_en$, $bkn_fl_wr_en$) is asserted in conjunction with the reference address for a single rising clock when a new access request is made.

Addresses are driven to the flash array in a flow-through fashion to minimize array access time. When no outstanding access is in progress, the platform flash controller drives addresses and asserts $bkn_fl_rd_en$ or $bkn_fl_wr_en$ and then may change to the next outstanding address in the next cycle.

Accesses are terminated under control of the appropriate read/write wait state control setting. Thus, the access time of the operation is determined by the settings of the wait state control fields. Access timing can be varied to account for the operating conditions of the device (frequency, voltage, temperature) by appropriately setting the fields in the programming model for either bank.

The platform flash controller also has the capability of extending the normal AHB access time by inserting additional wait states for reads and writes. This capability is provided to allow emulation of other memories that have different access time characteristics. The added wait state specifications are provided by bit 28 to bit 24 of Flash address ($haddr[28:24]$, see [Table 143](#) and [Table 144](#)). These wait states are applied in addition to the normal wait states incurred for flash accesses. Refer to [Section 18.2.17, “Wait state emulation”](#), for more details.

Prefetching of next sequential page is blocked when $haddr[28:24]$ is non-zero. Buffer hits are also blocked as well, regardless of whether the access corresponds to valid data in one of the page read buffers. These steps are taken to ensure that timing emulation is correct and that excessive prefetching is avoided. In addition, to prevent erroneous operation in certain rare cases, the buffers are invalidated on any non-sequential AHB access with a non-zero value on $haddr[28:24]$.

18.2.7 Access protections

The platform flash controller provides programmable configurable access protections for both read and write cycles from masters via the Platform Flash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in [Section 18.2.4.2.3, “Platform Flash Access Protection Register \(PFAPR\)”](#). Detection of a protection violation results in an error response from the platform flash controller on the AHB transfer.

18.2.8 Read cycles — buffer miss

Read cycles from the flash array are initiated by driving a valid access address on $bkn_fl_addr[23:0]$ and asserting $bkn_fl_rd_en$ for the required setup (and hold) time before (and after) the rising edge of $hclk$. The platform flash controller then waits for the programmed number of read wait states before sampling the read data on $bkn_fl_rdata[127:0]$. This data is normally stored in the least-recently updated page read buffer for bank0 in parallel with the requested data being forwarded to the AHB. For bank1, the data is captured in the page-wide temporary holding register as the requested data is forwarded to the AHB bus. Timing diagrams of basic read accesses from the flash array are shown in [Figure 138](#) through [Figure 141](#).

If the flash access was the direct result of an AHB transaction, the page buffer is marked as most-recently-used as it is being loaded. If the flash access was the result of a speculative prefetch to the next sequential line, it is first loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs.

18.2.9 Read cycles — buffer hit

Single cycle read responses to the AHB are possible with the platform flash controller when the requested read access was previously loaded into one of the bank0 page buffers. In these “buffer hit” cases, read data is returned to the AHB data phase with a 0 wait state response.

Likewise, the bank1 logic includes a single 32-bit temporary holding register and sequential accesses that “hit” in this register are also serviced with a 0 wait state response.

18.2.10 Write cycles

In a write cycle, address, write data, and control signals are launched off the same edge of hclk at the completion of the first AHB data phase cycle. Write cycles to the flash array are initiated by driving a valid access address on `bkn_fl_addr[23:0]`, driving write data on `bkn_fl_wdata[63:0]`, and asserting `bkn_fl_wr_en`. Again, the controller drives the address and control information for the required setup time before the rising edge of hclk, and provides the required amount of hold time. The platform flash controller then waits for the appropriate number of write wait states before terminating the write operation. On the cycle following the programmed wait state value, the platform flash controller asserts `hready_out` to indicate to the AHB port that the cycle has terminated.

18.2.11 Error termination

The platform flash controller follows the standard procedure when an AHB bus cycle is terminated with an ERROR response. First, the platform flash controller asserts `hresp[0]` and negates `hready_out` to signal an error has occurred. On the following clock cycle, the platform flash controller asserts `hready_out` and holds both `hresp[0]` and `hready_out` asserted until `hready_in` is asserted.

The first case that can cause an error response to the AHB is when an access is attempted by an AHB master whose corresponding Read Access Control or Write Access Control settings do not allow the access, thus causing a protection violation. In this case, the platform flash controller does not initiate a flash array access.

The second case that can cause an error response to the AHB is when an access is performed to the flash array and is terminated with a flash error response. See [Section 18.2.13, “Flash error response operation”](#). This may occur for either a read or a write operation.

The third case that can cause an error response to the AHB is when a write access is attempted to the flash array and is disallowed by the state of the `bkn_fl_ary_access` control input. This case is similar to case 1.

A fourth case involves an attempted read access while the flash array is busy doing a write (program) or erase operation if the appropriate read-while-write control field is programmed for this response. The 3-bit read-while-write control allows for immediate termination of an attempted read, or various stall-while-write/erase operations are occurring.

The platform flash controller can also terminate the current AHB access if `hready_in` is asserted before the end of the current bus access. While this circumstance should not occur, this does not result in an error condition being reported, as this behavior is initiated by the AHB. In this circumstance, the platform flash controller control state machine completes any flash array access in progress (without signaling the AHB) before handling a new access request.

18.2.12 Access pipelining

The platform flash controller does not support access pipelining since this capability is not supported by the flash array. As a result, the APC (Address Pipelining Control) field should typically be the same value as the RWSC (Read Wait State Control) field for best performance, that is, $BK_n_APC = BK_n_RWSC$. It cannot be less than the RWSC.

18.2.13 Flash error response operation

The flash array may signal an error response by asserting `bkn_fl_xfr_err` to terminate a requested access with an error. This may occur due to an uncorrectable ECC error, or because of improper sequencing during program/erase operations. When an error response is received, the platform flash controller does not update or validate a bank0 page read buffer nor the bank1 temporary holding register. An error response may be signaled on read or write operations. For more information on the specifics related to signaling of errors, including flash ECC, refer to subsequent sections in this chapter.

18.2.14 Bank0 page read buffers and prefetch operation

The logic associated with bank0 of the platform flash controller contains four 128-bit page read buffers that hold data read from the flash array. Each buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

The organization of each page buffer is described as follows in a pseudo-code representation. The hardware structure includes the buffer address and valid bit, along with 128 bits of page read data and several error flags.

```
struct {
    // bk0_page_buffer
    reg    addr[23:4]; // page address
    reg    valid;      // valid bit
    reg    rdata[127:0]; // page read data
    reg    xfr_error;  // transfer error indicator from flash array
    reg    multi_ecc_error; // multi-bit ECC error indicator from flash array
    reg    single_ecc_error; // single-bit correctable ECC indicator from flash array
} bk0_page_buffer[4];
```

For the general case, a page buffer is written at the completion of an error-free flash access and the valid bit asserted. Subsequent flash accesses that “hit” the buffer, that is, the current access address matches the address stored in the buffer, can be serviced in 0 AHB wait states as the stored read data is routed from the given page buffer back to the requesting bus master.

As noted in [Section 18.2.13, “Flash error response operation”](#), a page buffer is *not* marked as valid if the flash array access terminated with any type of transfer error. However, the result is that flash array accesses

that are tagged with a single-bit correctable ECC event are loaded into the page buffer and validated. For additional comments on this topic, see [Section 18.2.14.4, “Buffer invalidation”](#).

Prefetch triggering is controllable on a per-master and access-type basis. Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access to the platform flash controller may trigger a prefetch to the next sequential page of array data on the first idle cycle following the request. The access address is incremented to the next-higher 16-byte boundary, and a flash array prefetch is initiated if the data is not already resident in a page buffer. Prefetched data is always loaded into the least-recently-used buffer.

Buffers may be in one of six states, listed here in prioritized order:

1. Invalid—the buffer contains no valid data.
2. Used—the buffer contains valid data that has been provided to satisfy an AHB burst type read.
3. Valid—the buffer contains valid data that has been provided to satisfy an AHB single type read.
4. Prefetched—the buffer contains valid data that has been prefetched to satisfy a potential future AHB access.
5. Busy AHB—the buffer is currently being used to satisfy an AHB burst read.
6. Busy Fill—the buffer has been allocated to receive data from the flash array, and the array access is still in progress.

Selection of a buffer to be loaded on a miss is based on the following replacement algorithm:

1. First, the buffers are examined to determine if there are any invalid buffers. If there are multiple invalid buffers, the one to be used is selected using a simple numeric priority, where buffer 0 is selected first, then buffer 1, etc.
2. If there are no invalid buffers, the least-recently-used buffer is selected for replacement.

Once the candidate page buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the just-loaded buffer is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, the recently-used status is not changed. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control that trade off performance versus power. They are defined by the `Bx_Py_PFLM` (prefetch limit) register field. More aggressive prefetching increases power slightly due to the number of wasted (discarded) prefetches, but may increase performance by lowering average read latency.

In order for prefetching to occur, a number of control bits must be enabled. Specifically, the global buffer enable (`Bx_Py_BFE`) must be set, the prefetch limit (`Bx_Py_PFLM`) must be non-zero and either instruction prefetching (`Bx_Py_IPFE`) or data prefetching (`Bx_Py_DPFE`) enabled. Refer to [Section 18.2.4.2, “Registers description”](#), for a description of these control fields.

18.2.14.1 Instruction/data prefetch triggering

Prefetch triggering may be enabled for instruction reads via the Bx_Py_IPFE control field, while prefetching for data reads is enabled via the Bx_Py_DPFE control field. Additionally, the Bx_Py_PFLIM field must also be set to enable prefetching. Prefetches are never triggered by write cycles.

18.2.14.2 Per-master prefetch triggering

Prefetch triggering may be also controlled for individual bus masters. AHB accesses indicate the requesting master via the hmaster[3:0] inputs. Refer to [Section 18.2.4.2.3, “Platform Flash Access Protection Register \(PFAPR\)”](#) for details on these controls.

18.2.14.3 Buffer allocation

Allocation of the line read buffers is controlled via page buffer configuration (Bx_Py_BCFG) field. This field defines the operating organization of the four page buffers. The buffers can be organized as a “pool” of available resources (with all four buffers in the pool) or with a fixed partition between buffers allocated to instruction or data accesses. For the fixed partition, two configurations are supported. In one configuration, buffers 0 and 1 are allocated for instruction fetches and buffers 2 and 3 for data accesses. In the second configuration, buffers 0, 1, and 2 are allocated for instruction fetches and buffer 3 reserved for data accesses.

18.2.14.4 Buffer invalidation

The page read buffers may be invalidated under hardware or software control.

Any falling edge transition of the array’s bkn_fl_done signal causes the page read buffers to be marked as invalid. This input is negated by the flash array at the beginning of all program/erase operations as well as in certain other cases. Buffer invalidation occurs at the next AHB non-sequential access boundary, but does not affect a burst from a page read buffer in progress.

Software may invalidate the buffers by clearing the Bx_Py_BFE bit, which also disables the buffers. Software may then re-assert the Bx_Py_BFE bit to its previous state, and the buffers will have been invalidated.

One special case needing software invalidation relates to page buffer “hits” on flash data that was tagged with a single-bit ECC event on the original array access. Recall that the page buffer structure includes a status bit signaling the array access detected and corrected a single-bit ECC error. On all subsequent buffer hits to this type of page data, a single-bit ECC event is signaled by the platform flash controller. Depending on the specific hardware configuration, this reporting of a single-bit ECC event may generate an ECC alert interrupt. In order to prevent repeated ECC alert interrupts, the page buffers need to be invalidated by software after the first notification of the single-bit ECC event.

Finally, the buffers are invalidated by hardware on any non-sequential access with a non-zero value on haddr[28:24] to support wait state emulation.

18.2.15 Bank1 temporary holding register

Recall the bank1 logic within the flash includes a single 128-bit data register, used for capturing read data. Since this bank does not support prefetching, the read data for the referenced address is bypassed directly back to the AHB data bus. The page is also loaded into the temporary data register and subsequent accesses to this page can hit from this register, if it is enabled (B1_Py_BFE).

The organization of the temporary holding register is described as follows, in a pseudo-code representation. The hardware structure includes the buffer address and valid bit, along with 128 bits of page read data and several error flags and is the same as an individual bank0 page buffer.

```
struct {
    // bk1_page_buffer
    reg    addr[23:4]; // page address
    reg    valid;      // valid bit
    reg    rdata[127:0]; // page read data
    reg    xfr_error;  // transfer error indicator from flash array
    reg    multi_ecc_error; // multi-bit ECC error indicator from flash array
    reg    single_ecc_error; // single-bit correctable ECC indicator from flash array
}
bk1_page_buffer;
```

For the general case, a temporary holding register is written at the completion of an error-free flash access and the valid bit asserted. Subsequent flash accesses that “hit” the buffer, that is, the current access address matches the address stored in the temporary holding register, can be serviced in 0 AHB wait states as the stored read data is routed from the temporary register back to the requesting bus master.

The contents of the holding register are invalidated by the falling edge transition of `bk1_fl_done` and on any non-sequential access with a non-zero value on `haddr[28:24]` (to support wait state emulation) in the same manner as the bank0 page buffers. Additionally, the B1_Py_BFE register bit can be cleared by software to invalidate the contents of the holding register.

As noted in [Section 18.2.13, “Flash error response operation”](#), the temporary holding register is *not* marked as valid if the flash array access terminated with any type of transfer error. However, the result is that flash array accesses that are tagged with a single-bit correctable ECC event are loaded into the temporary holding register and validated. Accordingly, one special case needing software invalidation relates to holding register “hits” on flash data that was tagged with a single-bit ECC event. Depending on the specific hardware configuration, the reporting of a single-bit ECC event may generate an ECC alert interrupt. In order to prevent repeated ECC alert interrupts, the page buffers need to be invalidated by software after the first notification of the single-bit ECC event.

The bank1 temporary holding register effectively operates like a single page buffer.

18.2.16 Read-While-Write functionality

The platform flash controller supports various programmable responses for read accesses while the flash is busy performing a write (program) or erase operation. For all situations, the platform flash controller uses the state of the flash array’s `bk n _fl_done` output to determine if it is busy performing some type of high-voltage operation, namely, if `bk n _fl_done = 0`, the array is busy.

Specifically, there are two 3-bit read-while-write (BK n _RWWC) control register fields that define the platform flash controller’s response to these types of access sequences. There are five unique responses

that are defined by the BK_n_RWWC setting: one immediately reports an error on an attempted read, and four settings that support various stall-while-write capabilities. Consider the details of these settings.

- $BK_n_RWWC = 0b0xx$
 - For this mode, any attempted flash read to a busy array is immediately terminated with an AHB error response and the read is blocked in the controller and not seen by the flash array.
- $BK_n_RWWC = 0b111$
 - This defines the basic stall-while-write capability and represents the default reset setting. For this mode, the platform flash controller module stalls any read reference until the flash has completed its program/erase operation. If a read access arrives while the array is busy or if a falling-edge on bkn_fl_done occurs while a read is still in progress, the AHB data phase is stalled by negating $hready_out$ and saving the address and attributes into holding registers. Once the array has completed its program/erase operation, the platform flash controller uses the saved address and attribute information to create a pseudo address phase cycle to “retry” the read reference and sends the registered information to the array as $bkn_fl_rd_en$ is asserted. Once the retried address phase is complete, the read is processed normally and once the data is valid, it is forwarded to the AHB bus and $hready_out$ negated to terminate the system bus transfer.
- $BK_n_RWWC = 0b110$
 - This setting is similar to the basic stall-while-write capability provided when $BK_n_RWWC = 0b111$ with the added ability to generate a notification interrupt if a read arrives while the array is busy with a program/erase operation. There are two notification interrupts, one for each bank.
- $BK_n_RWWC = 0b101$
 - Again, this setting provides the basic stall-while-write capability with the added ability to terminate any program/erase operation if a read access is initiated. For this setting, the read request is captured and retried as described for the basic stall-while-write, plus the program/erase operation is terminated by the platform flash controller’s assertion of the bkc_fl_abort signal. The bkn_fl_abort signal remains asserted until bkn_fl_done is driven high. For this setting, there are no notification interrupts generated.
- $BK_n_RWWC = 0b100$
 - This setting provides the basic stall-while-write capability with the ability to terminate any program/erase operation if a read access is initiated plus the generation of a termination notification interrupt. For this setting, the read request is captured and retried as described for the basic stall-while-write, the program/erase operation is terminated by the platform flash controller’s assertion of the bkn_fl_abort signal and a termination notification interrupt generated. There are two termination notification interrupts, one for each bank.

As detailed above, there are a total of four interrupt requests associated with the stall-while-write functionality. These interrupt requests are captured as part of MCM’s Interrupt Register and logically summed together to form a single request to the interrupt controller.

Table 142. Platform flash controller stall-while-write interrupts

MIR[n]	Interrupt description
MCM.MIR[7]	Platform flash bank0 termination notification, MIR[FB0AI]
MCM.MIR[6]	Platform flash bank0 stall notification, MIR[FB0SI]
MCM.MIR[5]	Platform flash bank1 termination notification, MIR[FB1AI]
MCM.MIR[4]	Platform flash bank1 stall notification, MIR[FB1S1]

For example timing diagrams of the stall-while-write and terminate-while-write operations, see [Figure 142](#) and [Figure 143](#) respectively.

18.2.17 Wait state emulation

Emulation of other memory array timings are supported by the platform flash controller on read cycles to the flash. This functionality may be useful to maintain the access timing for blocks of memory that were used to overlay flash blocks for the purpose of system calibration or tuning during code development.

The platform flash controller inserts additional wait states according to the values of `haddr[28:24]`, where `haddr` represents the Flash address. When these inputs are non-zero, additional cycles are added to AHB read cycles. Write cycles are not affected. In addition, no page read buffer prefetches are initiated, and buffer hits are ignored.

[Table 143](#) and [Table 144](#) show the relationship of `haddr[28:24]` to the number of additional primary wait states. These wait states are applied to the initial access of a burst fetch or to single-beat read accesses on the AHB system bus.

Note that the wait state specification consists of two components: `haddr[28:26]` and `haddr[25:24]` and effectively extends the flash read by $(8 \times \text{haddr}[25:24] + \text{haddr}[28:26])$ cycles.

Table 143. Additional wait state encoding

Memory address <code>haddr[28:26]</code>	Additional wait states
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

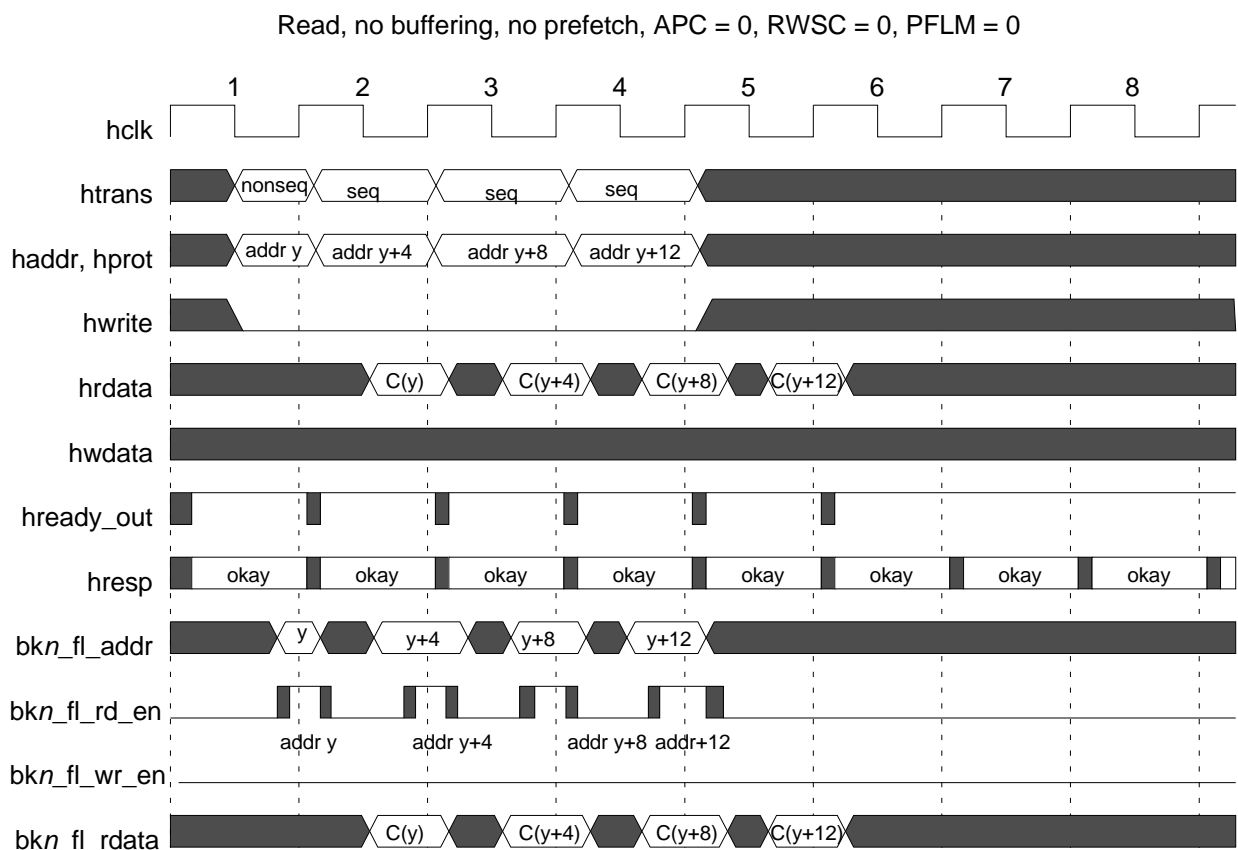
[Table 144](#) shows the relationship of `haddr[25:24]` to the number of additional wait states. These are applied in addition to those specified by `haddr[28:26]` and thus extend the total wait state specification capability.

Table 144. Extended additional wait state encoding

Memory address haddr[25:24]	Additional wait states (added to those specified by haddr[28:26])
00	0
01	8
10	16
11	24

18.2.18 Timing diagrams

Since the platform flash controller is typically used in platform configurations with a cacheless core, the operation of the processor accesses to the platform memories, for example flash and SRAM, plays a major role in the overall system performance. Given the core/platform pipeline structure, the platform's memory controllers (PFLASH, PRAM) are designed to provide a 0 wait state data phase response to maximize processor performance. The following diagrams illustrate operation of various cycle types and responses referenced earlier in this chapter including stall-while-read (Figure 142) and terminate-while-read (Figure 143) diagrams.


Figure 138. 1-cycle access, no buffering, no prefetch

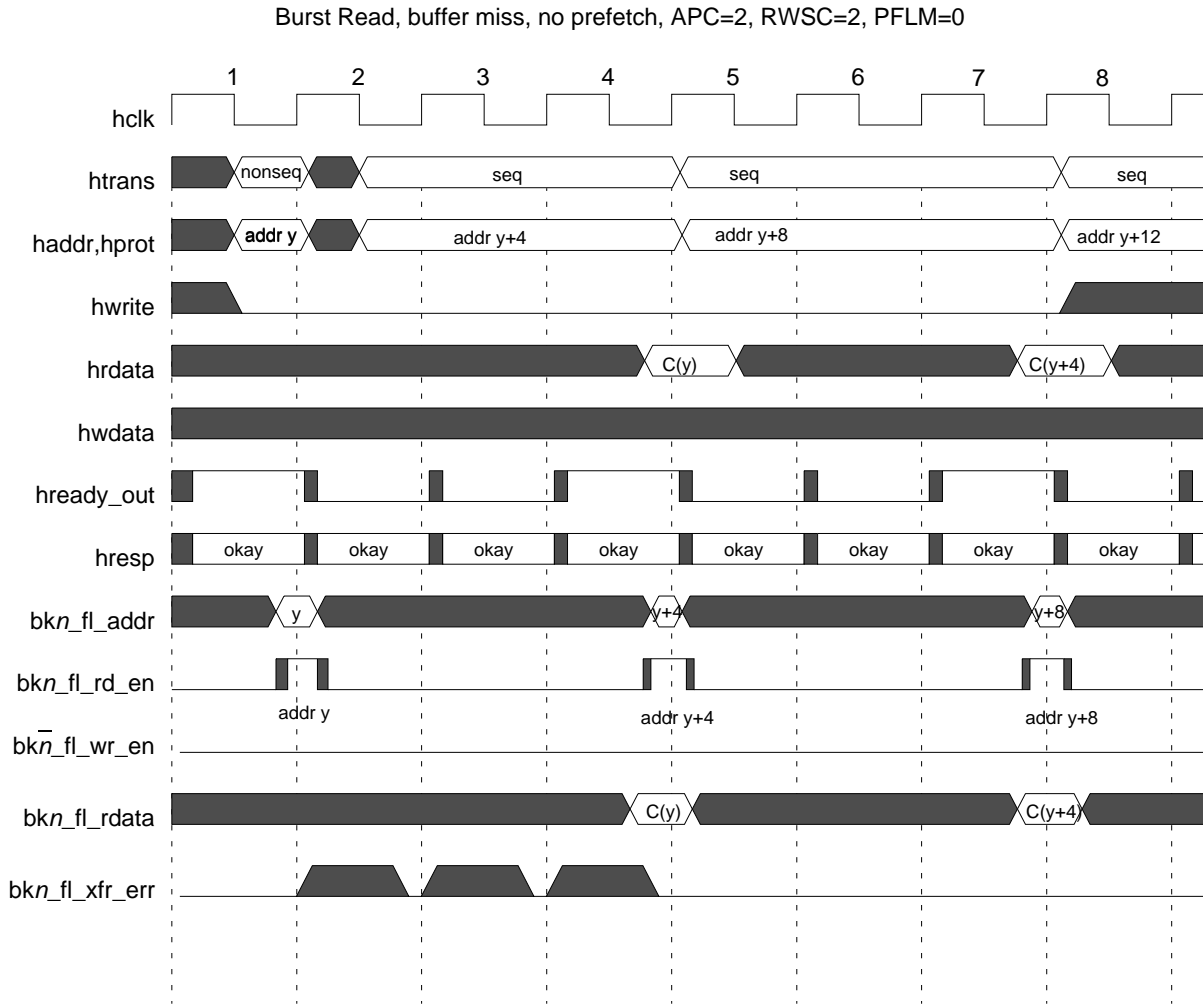


Figure 139. 3-cycle access, no prefetch, buffering disabled

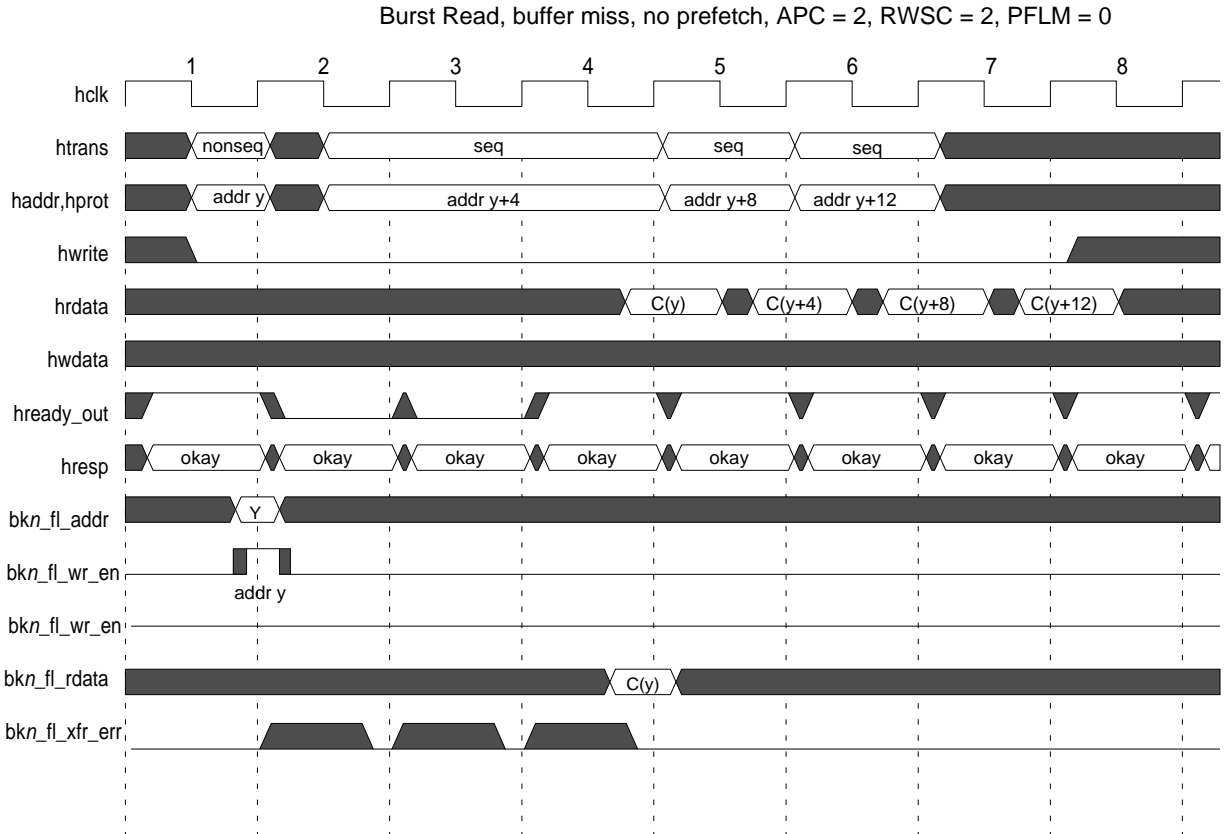


Figure 140. 3-cycle access, no prefetch, buffering enabled

Burst Read, buffer miss, prefetch, APC = 2, RWSC = 2, PFLM = 2

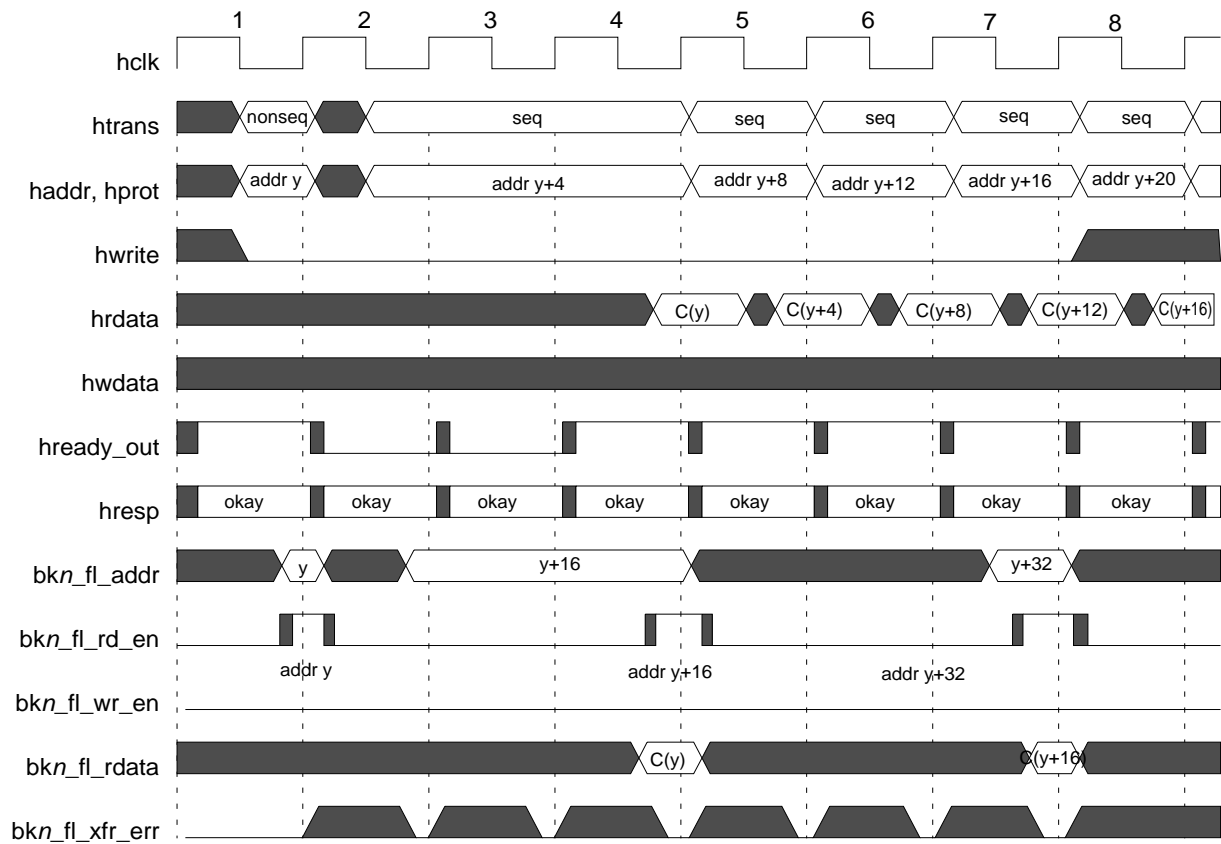


Figure 141. 3-cycle access, prefetch and buffering enabled

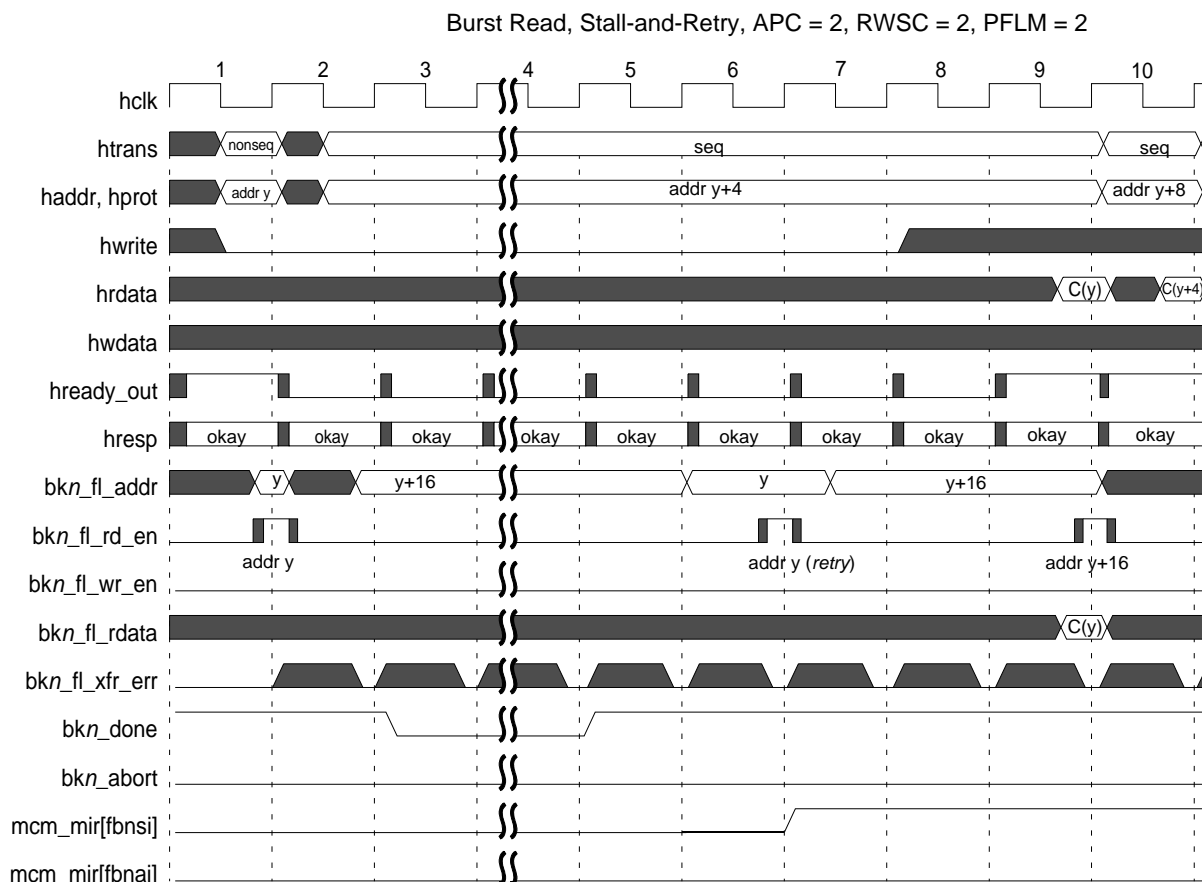


Figure 142. 3-cycle access, stall-and-retry with BK_n RWWC = 11x

As shown in [Figure 142](#), the 3-cycle access to address y is interrupted when an operation causes the `bkn_done` signal to be negated, signaling that the array bank is busy with a high-voltage program or erase event. Eventually, this array operation completes (at the end of cycle 4) and `bkn_done` returns to a logical 1. In cycle 6, the platform flash controller module retries the read to address y that was interrupted by the negation of `bkn_done` in cycle 3. Note that throughout cycles 2–9, the AHB bus pipeline is stalled with a read to address y in the AHB data phase and a read to address $y + 4$ in the address phase. Depending on the state of the least-significant-bit of the BK_n RWWC control field, the hardware may also signal a stall notification interrupt (if BK_n RWWC = 110). The stall notification interrupt is shown as the optional assertion of MCM’s MIR[FB_nSI] (flash bank n stall interrupt).

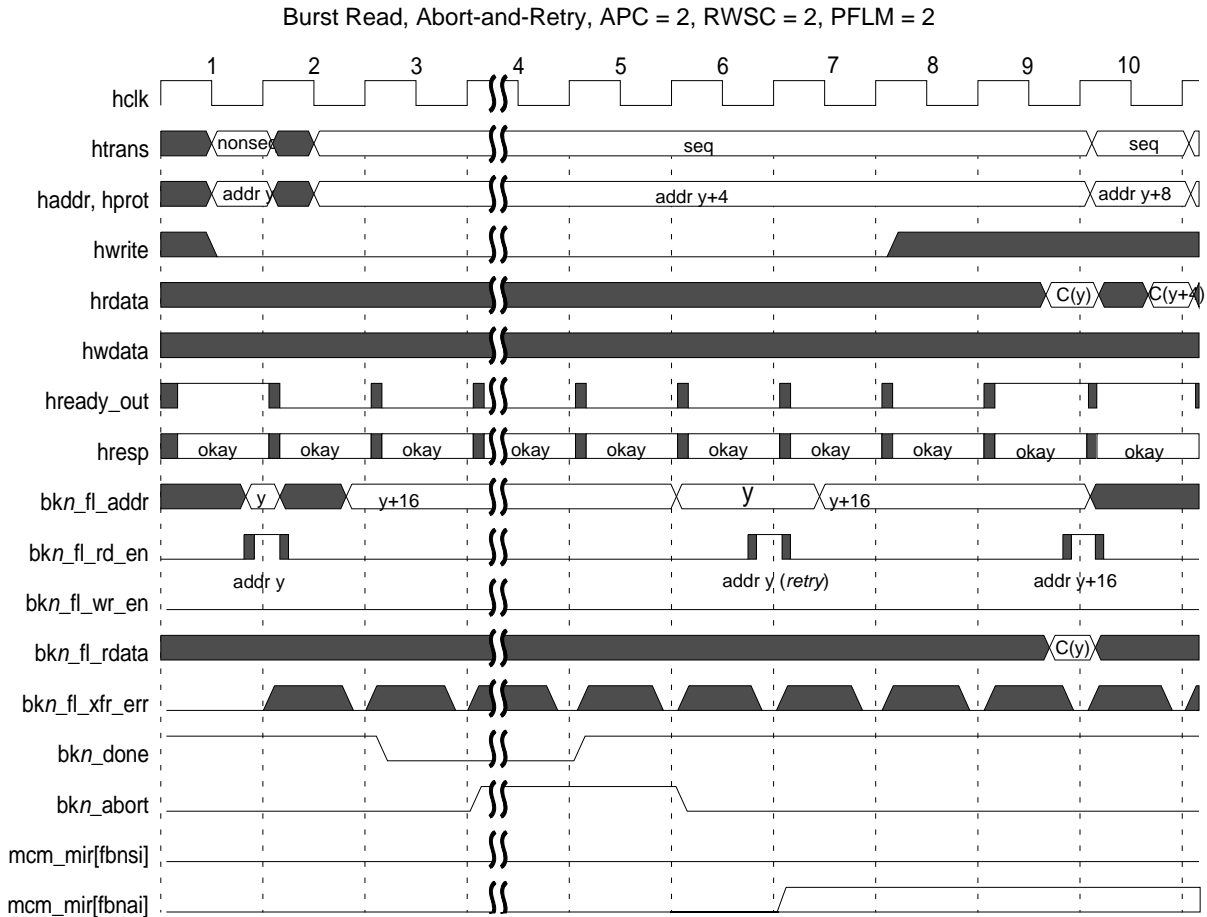


Figure 143. 3-cycle access, terminate-and-retry with BK_n RWWC = 10x

Figure 143 shows the terminate-while-write timing diagram. In this example, the 3-cycle access to address *y* is interrupted when an operation causes the *bkn_done* signal to be negated, signaling that the array bank is busy with a high-voltage program or erase event. Based on the setting of BK_n RWWC, once the *bkn_done* signal is detected as negated, the platform flash controller asserts *bkn_abort*, which forces the flash array to cancel the high-voltage program or erase event. The array operation completes (at the end of cycle 4) and *bkn_done* returns to a logical 1. It should be noted that the time spent in cycle 4 for Figure 143 is considerably less than the time in the same cycle in Figure 142 (because of the terminate operation). In cycle 6, the platform flash controller module retries the read to address *y* that was interrupted by the negation of *bkn_done* in cycle 3. Note that throughout cycles 2–9, the AHB bus pipeline is stalled with a read to address *y* in the AHB data phase and a read to address *y*+4 in the address phase. Depending on the state of the least-significant-bit of the BK_n RWWC control field, the hardware may also signal an termination notification interrupt (if BK_n RWWC = 100). The stall notification interrupt is shown as the optional assertion of MCM's MIR[FBnAI] (flash bank *n* termination interrupt).

18.3 Code Flash Memory (C90LC)

18.3.1 Overview

The primary function of the Flash Module is to serve as electrically programmable and erasable Non-Volatile Memory.

NV Memory may be used for instruction and/or data storage.

The Module is a Non-Volatile solid-state silicon memory device consisting of blocks (called also sectors) of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The Flash Module is arranged as two functional units: the Flash Core and the Memory Interface.

The Flash Core is composed of arrayed Non-Volatile storage elements, sense amplifiers, row decoders, column decoders and charge pumps. The arrayed storage elements in the Flash Core are sub-divided into physically separate units referred to as blocks (or sectors).

The Memory Interface contains the registers and logic which control the operation of the Flash Core. The Memory Interface is also the interface between the Flash Module and a Bus Interface Unit (BIU) and may contain the ECC logic and redundancy logic.

A BIU connects the Flash Module to a system bus, and contains all system level customization required for the SoC application. The Flash Module is generic and requires a BIU to configure it for different SoC applications. A BIU is not included as a part of the Flash Module.

18.3.2 Features

- Good Access Time
- High Read parallelism (128 bits)
- Error Correction Code (SEC-DED) to enhance Data Retention
- Double Word Program (64 bits)
- Sector Erase
- Single Bank: Read-While-Modify not available
- Erase Suspend available (Program Suspend not available)
- Software programmable Program/Erase Protection to avoid unwanted writings
- Censored Mode against piracy
- Usable as main Code Memory of the device: Shadow Sector available

18.3.3 Block Diagram

The Flash Macrocell contains one Matrix Module, composed by a Single Bank: Bank 0, normally used for Code storage. No Read-While-Modify operations are possible.

The Modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are given through a User Registers Interface.

The read data bus is 128 bits wide, while the Flash registers are on a separate bus 32 bits wide. The High Voltages needed for Program/Erase operations are internally generated.

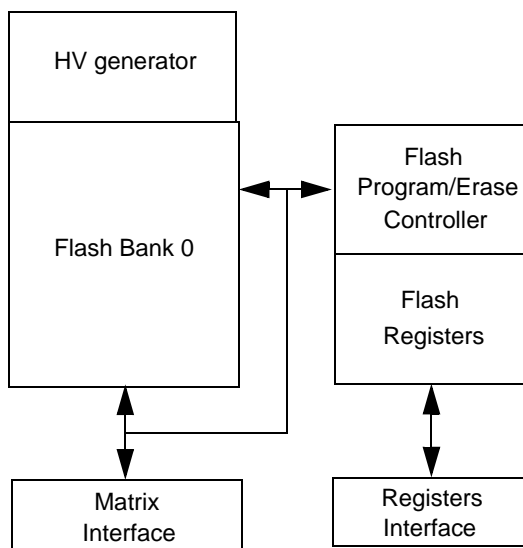


Figure 144. Flash Macrocell Structure

18.3.4 Functional Description

18.3.4.1 Macrocell Structure

The Flash Macrocell is designed for use in embedded MCU/SoC applications which require high density Non-Volatile Memories with high speed read access. The Flash Module is addressable by Word (32 bits) or Double Word (64 bits) for program, and page (128 bits) for read. Reads done to the Flash always return 128 bits, although read page buffering may be done in the platform BIU.

Each read of the Flash Module retrieves a page, or 4 consecutive words (128 bits) of information. The address for each word retrieved within a page differ from the other addresses in the page only by address bits (3:2). The Flash page read architecture easily supports both cache and burst mode at the BIU level for high speed read application.

The Flash Module supports fault tolerance through Error Correction Code (ECC) and/or error detection. The ECC implemented within the Flash Module will correct single bit failures and detect double bit failures.

The Flash Module uses an embedded hardware algorithm implemented in the Memory Interface to program and erase the Flash Core. Control logic that works with the software block enables, and software lock mechanisms, is included in the embedded hardware algorithm to guard against accidental program/erase. The hardware algorithm performs the steps necessary to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the Flash Module reads as logic level 0 (or low). An erased bit in the Flash Module reads as logic level 1 (or high). Program and erase of the Flash Module requires multiple system clock cycles to complete. The erase sequence may be suspended. The program and erase sequences may be aborted.

18.3.5 Code flash sectorization

The Flash Module supports total memory sizes ranging from 32 KB to 512 KB of User Memory, plus 16 KB of Test Memory (a portion of which is One-Time Programmable by the User). Optionally an extra sector of 8 or 16 KB can be available as Shadow space.

- There are three User Address Spaces: Low, Mid and High Address Space.
- Low Address Space must always be present and be up to 256 KB in size.
- Mid Address Space can be present and be up to 256 KB in size.
- High Address Space is normally always empty, but it can be present in case there are holes in the address space. High Address Space may extend up to 1.5 MB in the address mapping.

In any case the total size of the Flash Module will be not greater than 512 KB and the maximum number of blocks cannot exceed 16, included Test and eventually Shadow Sector.

There are five sizes of blocks available to the User in the Flash Core: 128 KB, 64 KB, 32 KB, 16 KB, 8 KB. These blocks can be mapped anywhere in the Low, Mid and High address spaces, provided that the total Flash Memory size is not greater than 512 KB.

The Flash Module is composed by a single Bank (Bank 0): Read-While-Modify is not supported.

Bank 0 of the 544 KB Flash macrocell is divided in 10 sectors. Bank 0 contains also a reserved sector named TestFlash in which some One Time Programmable User data are stored. Besides Bank 0 contains also a Shadow Sector in which User erasable configuration values can be stored.

Table 145. 544 KB Code flash module sectorization

Bank	Sector	Addresses	Size	Address space
B0	B0F0	0x0000_0000–0x0000_3FFF	16 KB	Low Address Space
B0	B0F1	0x0000_4000–0x0000_7FFF	16 KB	Low Address Space
B0	B0F2	0x0000_8000–0x0000_FFFF	32 KB	Low Address Space
B0	B0F3	0x0001_0000–0x0001_7FFF	32 KB	Low Address Space
B0	B0F4	0x0001_8000–0x0001_BFFF	16 KB	Low Address Space
B0	B0F5	0x0001_C000–0x0001_FFFF	16 KB	Low Address Space
B0	B0F6	0x0002_0000–0x0002_FFFF	64 KB	Low Address Space
B0	B0F7	0x0003_0000–0x0003_FFFF	64 KB	Low Address Space
B0	B0F8	0x0004_0000–0x0005_FFFF	128 KB	Mid Address Space
B0	B0F9	0x0006_0000–0x0007_FFFF	128 KB	Mid Address Space

Table 145. 544 KB Code flash module sectorization

Bank	Sector	Addresses	Size	Address space
B0	Reserved	0x0008_0000–0x001F_FFFF	1536 KB	High Address Space
B0	B0SH	0x0020_0000–0x0020_3FFF	16 KB	Shadow Address Space
B0	B0TF	0x0040_0000–0x0040_3FFF	16 KB	Test Address Space

The Flash Module is divided into blocks also to implement independent Erase/Program protection. A software mechanism is provided to independently lock/unlock each block in low, mid and high address space against program and erase.

18.3.5.1 Test Flash Block

The TestFlash block exists outside the normal address space and is programmed, erased and read independently of the other blocks. The independent TestFlash block is included also to support systems which require Non-Volatile Memory for security and/or to store system initialization information.

A section of the TestFlash is reserved to store the Non Volatile informations related to Redundancy, Configuration and Protection.

Due to this special usage, the TestFlash sector is not affected by the Column Redundancy. The ECC, on the contrary, is applied also to TestFlash.

The usage of reserved TestFlash sector is detailed in the following table.

Table 146. TestFlash Structure

Name	Description	Addresses	Size
	User OTP Area	0x400000 to 0x401FFF	8192 byte
	Reserved	0x402000 to 0x403CFF	7424 byte
	User Reserved	0x403D00 to 0x403DE7	232 byte
NVLML	NV Low/Mid address space block Locking reg	0x403DE8 to 0x403DEF	8 byte
NVHBL	Non Volatile High address space Block Locking reg	0x403DF0 to 0x403DF7	8 byte
NVSLL	NV Secondary Low/mid add space block Lock reg	0x403DF8 to 0x403DFF	8 byte
	User Reserved	0x403E00 to 0x403EFF	256 byte
	Reserved	0x403F00 to 0x403FFF	256 byte

The Test Flash block can be enabled by the BIU.

When the Test space is enabled, all the operations are mapped to the Test block.

User Mode program of the test block are enabled only when MCR.PEAS is high, also if the Shadow block is available.

The Test Flash block may be locked/unlocked against program by using the LML.TSLK and SLL.STSLK registers. Erase of Test Flash block is always locked in user mode.

Program of the TestFlash block has similar restriction as the array in terms of how ECC is calculated. Only one program is allowed per 64 bit ECC segment, unless ECC evaluation is disabled on TestFlash block (SoC dependent).

The TestFlash block contains specified data that are needed for Flash Macrocell or SoC features.

The first 8KB of TestFlash block may be used for user defined functions (possibly to store boot code, other configuration words or factory process codes). Locations of the TestFlash block marked as reserved cannot be programmed by the User application.

18.3.5.2 Shadow block

A Shadow block is present in the 544 KB Flash Macrocell. The Shadow block can be enabled by the BIU.

When the Shadow space is enabled, all the operations are mapped to the Shadow block.

User Mode program and erase of the shadow block are enabled only when MCR.PEAS is high.

The Shadow block may be locked/unlocked against program or erase by using the LML.TSLK and SLL.STSLK registers.

Program of the Shadow block has similar restriction as the array in terms of how ECC is calculated. Only one program is allowed per 64 bit ECC segment between erases, unless ECC evaluation is disabled on Shadow block (SoC dependent).

Erase of the Shadow block is done similarly as an array erase.

The Shadow block contains specified data that are needed for SoC features.

The first 8KB of Shadow block may be used for user defined functions (possibly to store boot code, other configuration words or factory process codes).

The usage of Shadow sector is detailed in the following table:

Table 147. Shadow Sector Structure

Addresses	Name	Description	Size
0x200000 to 0x203DCF		User Area	15824 byte
0x203DD0 to 0x203DD7		Reserved	8 byte
0x203DD8 to 0x203DDF	NVPWD0-1	Non Volatile private censorship PassWorD 0-1 reg	8 byte
0x203DE0 to 0x203DE7	NVSCI0-1	Non Volatile System Censorship Information 0-1	8 byte
0x203DE8 to 0x203DFF		Reserved	24 byte
0x203E00 to 0x203E0F	NVBIU2-3	Non Volatile Bus Interface Unit 2-3 regs	16 byte
0x203E10 to 0x203E17		Reserved	8 byte
0x203E18 – 0x203E1F	NVUSRO	Non Volatile USer Options register	8 byte
0x203E20 – 0x203FFF		Reserved	480 byte

18.3.5.3 User Mode Operation

In User Mode the Flash Module may be read and written (register writes and interlock writes), programmed or erased.

The default state of the Flash Module is read. The main, shadow and test address space can be read only in the read state. The Flash registers are always available for read, also when the Module is in disable mode (except few documented registers).

The Flash Module enters the read state on reset. The Module is in the read state under two sets of conditions:

- The read state is active when the Module is enabled (User Mode Read)
- The read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend).

NOTE

No Read-While-Modify is available.

Flash Core reads return 128 bits (1 Page = 2 Double Words).

Registers reads return 32 bits (1 Word).

Flash Core reads are done through the Bus Interface Unit.

In many cases the BIU will do “read page buffering” to allow sequential reads to be done with higher performance. This could provide Data Coherency issue that must be handled with software. Data Coherency may be an issue after a program or an erase operation, as well as Shadow or Test block operations.

Registers reads to unmapped register address space will return all 0's.

Registers writes to unmapped register address space will have no effect.

Array reads attempted to invalid locations will result in indeterminate data. Invalid locations occur when addressing is done to blocks that do not exist in non 2^n array sizes.

Interlock writes attempted to invalid locations, will result in an interlock occurring, but attempts to program these blocks will not occur since they are forced to be locked. Erase will occur to selected and unlocked blocks even if the interlock write is to an invalid location.

Simultaneous Read cycle on the Flash Matrix and Read/Write cycles on the Registers are possible. On the contrary Registers Read/Write accesses simultaneous to a Flash Matrix interlock write are forbidden.

Chip Select, Write Enable, Addresses and Data Input of Registers are not internally latched and must be kept stable by the CPU for all the read/write access that lasts 2 clock cycles.

18.3.5.4 Reset

A reset is the highest priority operation for the Flash Module and terminates all other operations.

The Flash Module uses reset to initialize register and status bits to their default reset values.

If the Flash Module is executing a Program or Erase operation (MCR.PGM = 1 or MCR.ERS = 1) and a reset is issued, the operation will be suddenly terminated and the module will disable the high voltage logic without damage to the high voltage circuits. Reset terminates all operations and forces the Flash Module

into User Mode ready to receive accesses.

Reset and power-off must not be used as a systematic way to terminate a Program or Erase operation.

After reset is negated, read register access may be done, although it should be noted that registers that require updating from shadow information, or other inputs, may not read updated values until MCR.DONE transitions. MCR.DONE may be polled to determine if the Flash Module has transitioned out of reset. Notice that the registers cannot be written until MCR.DONE is high.

18.3.5.5 Disable Mode (Power-Down)

The Disable (or Power-Down) Mode allows to turn-off all Flash DC current sources, so that all power dissipation is due only to leakage in this mode.

In Disable Mode no reads from or write to the Module are possible.

The User may not read some registers (UMISR0-4, UT1-2 and part of UT0) until the Disable Mode is exited. On the contrary write access is locked on all the registers in Disable Mode.

When enabled the Flash Module returns to its pre-disable state in all cases unless in the process of executing an erase high voltage operation at the time of disable.

If the Flash Module is disabled during an erase operation, MCR.ESUS bit is set to 1. The User may resume the erase operation at the time the Module is enabled by clearing MCR.ESUS bit. MCR.EHV must be high to resume the erase operation.

If the Flash Module is disabled during a program operation, the operation will be in any case completed and the Disable Mode will be entered only after the programming end.

If the Flash Macrocell is put in Power-Down Mode and the Vector Table remain mapped in the Flash Address space, the User must take care than the Flash Macrocell will strongly increase the interrupt response time by adding several Wait States.

It is forbidden to enter in Sleep Mode when the Disable Mode is active.

18.3.5.6 Sleep Mode (Low Power Mode)

The Sleep Mode turns-off most of the DC current sources within the Flash Module. Wake-up time from sleep is faster than wake-up time from disable mode.

In Sleep Mode no reads from or write to the Module are possible.

The User may not read some registers (UMISR0-4, UT1-2 and part of UT0) until the Sleep Mode is exited. On the contrary write access is locked on all the registers in Sleep Mode.

When exiting from sleep mode the Flash Module returns to its pre-sleep state in all cases unless in the process of executing an erase high voltage operation at the time of sleep entering.

If the Flash Module is put in sleep during an erase operation, MCR.ESUS bit is set to 1. The User may resume the erase operation at the time the Module is exited from sleep by clearing MCR.ESUS bit. MCR.EHV must be high to resume the erase operation.

If the Flash Module is put in sleep during a program operation, the operation will be in any case completed and the Sleep Mode will be entered only after the programming end.

It is forbidden to enter in Disable Mode when the Sleep Mode is active.

18.3.6 Registers Description

The Flash User registers represents the communication interface between the host CPU and the FPEC. me register bits (command bits) are read/write for the CPU and read-only for the FPEC.

Some other register bits (status bits) are read/write for the FPEC and read-only for the CPU.

Table 148. Code flash module registers

Address Offset	Register Name	Reset Value
0x0000	Module Configuration Register (MCR)	0x02700600
0x0004	Low/Mid address space block Locking reg (LML)	0x00XX00XX
0x0008	High address space Block Locking reg (HBL)	0x00000000
0x000C	Secondary Low/mid address space block Lock reg (SLL)	0x00XX00XX
0x0010	Low/Mid address space block Select reg (LMS)	0x00000000
0x0014	High address space Block Select reg (HBS)	0x00000000
0x0018	ADress Register (ADR)	0x00000000
0x001C	Bus Interface Unit reg 0 (BIU0) ¹	0XXXXXXXXX
0x0020	Bus Interface Unit reg 1 (BIU1) ¹	0XXXXXXXXX
0x0024	Bus Interface Unit reg 2 (BIU2) ²	0XXXXXXXXX
0x002C	Reserved	—
0x003C	User Test reg 0 (UT0)	0x00000001
0x0040	User Test reg 1 (UT1)	0x00000000
0x0044	User Test reg 2 (UT2)	0x00000000
0x0048	User Multiple Input Signature Reg 0 (UMISR0)	0x00000000
0x004C	User Multiple Input Signature Reg 1 (UMISR1)	0x00000000
0x0050	User Multiple Input Signature Reg 2 (UMISR2)	0x00000000
0x0054	User Multiple Input Signature Reg 3 (UMISR3)	0x00000000
0x0058	User Multiple Input Signature Reg 4 (UMISR4)	0x00000000

¹ Bus Interface Unit register 0 and 1 (BIU0 and BIU1) are same as [Section 18.2.4.2.1, “Platform Flash Configuration Register 0 \(PFCR0\)”](#) and [Section 18.2.4.2.2, “Platform Flash Configuration Register 1 \(PFCR1\)”](#).

² Bus Interface Unit register 2 (BIU2) is same as [Section 18.2.4.2.3, “Platform Flash Access Protection Register \(PFAPR\)”](#)

In the following some Non Volatile Registers are described. Please notice that such entities are not Flip-Flops, but locations of TestFlash or Shadow sectors with a special meaning.

During the Flash Initialization phase, the FPEC reads these Non Volatile Registers and update their related Volatile Registers. When the FPEC detects ECC double errors in these special locations, it behaves in the following way:

- In case of a failing system locations (configurations, device options, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged .
- In case of failing user locations (protections, censorship, BIU, ...), the Volatile Registers are filled with all '1's and the Flash initialization ends setting low the PEG bit of MCR.

In this section, the following abbreviations are used.

Table 149. Abbreviations

Case	Abbrev.	Description
read/write	rw	The software can read and write to these bits.
read/clear	rc	The software can read and clear to these bits.
read-only	r	The software can only read these bits.
write-only	w	The software should only write to these bits.

18.3.6.1 Module Configuration Register (MCR)

The Module Configuration Register enables and monitors all the modify operations of each flash module. Identical MCRs are provided in the code flash and the data flash blocks.

Address: Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	EDC	0	0	0	0	SIZE2	SIZE1	SIZE0	0	LAS2	LAS1	LAS0	0	0	0	MAS	
W	r1c																
Reset	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	EER	RWE	0	0	PEAS	DONE	PEG	0	0	0	0	0	PGM	PSUS	ERS	ESUS	EHV
W	r1c	r1c															
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Figure 145. Module Configuration Register (MCR)

Table 150. MCR field descriptions

Field	Description
EDC	<p>ECC Data Correction</p> <p>EDC provides information on previous reads. If a ECC Single Error detection and correction occurs, the EDC bit is set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user.</p> <p>In the event of a ECC Double Error detection, this bit is not set.</p> <p>If EDC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EDC) were not corrected through ECC.</p> <p>Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect.</p> <p>0 Reads are occurring normally.</p> <p>1 An ECC Single Error occurred and was corrected during a previous read.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>

Table 150. MCR field descriptions (continued)

Field	Description
SIZE[2:0]	<p>Array space SIZE 2–0</p> <p>The value of SIZE field depends on the size of the flash module:</p> <p>000 128 KB 001 256 KB 010 512 KB 011 Reserved (1024 KB) 100 Reserved (1536 KB) 101 Reserved (2048 KB) 110 64 KB (the value for the device in the data flash module) 111 Reserved</p> <p>Note: The value for this bitfield is different between the code and data flash modules.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
LAS[2:0]	<p>Low Address Space 2–0</p> <p>The value of the LAS field corresponds to the configuration of the Low Address Space:</p> <p>000 Reserved 001 Reserved 011 Reserved 100 Reserved 101 Reserved 110 4 × 16 KB (the value for the device in the data flash module)</p> <p>Note: The value for this bitfield is different between the code and data flash modules.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
MAS	<p>Mid Address Space</p> <p>The value of the MAS field corresponds to the configuration of the Mid Address Space:</p> <p>0 2 × 128 KB 1 Reserved</p>
EER	<p>ECC Event Error</p> <p>EER provides information on previous reads. When an ECC Double Error detection occurs, the EER bit is set to 1.</p> <p>This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user.</p> <p>In the event of a ECC Single Error detection and correction, this bit will not be set.</p> <p>If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) were correct.</p> <p>Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect.</p> <p>0 Reads are occurring normally. 1 An ECC Double Error occurred during a previous read.</p>

Table 150. MCR field descriptions (continued)

Field	Description
RWE	<p>Read-while-Write event Error</p> <p>RWE provides information on previous reads when a Modify operation is on going. If a RWW Error occurs, the RWE bit is set to 1. Read-While-Write Error means that a read access to the flash module has occurred while the FPEC was performing a program or Erase operation or an Array Integrity Check.</p> <p>This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the user.</p> <p>If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) were correct.</p> <p>Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect.</p> <p>0 Reads are occurring normally. 1 A RWW Error occurred during a previous read.</p> <p>Note: If stall/terminate-while-write is used, the software should ignore the setting of the RWE flag and should clear this flag after each erase operation. If stall/terminate-while-write is not used, software can handle the RWE error normally.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
PEAS	<p>Program/Erase Access Space</p> <p>PEAS indicates which space is valid for program and Erase operations: main array space or shadow/test space.</p> <p>PEAS = 0 indicates that the main address space is active for all flash module program and erase operations.</p> <p>PEAS = 1 indicates that the test or shadow address space is active for program and erase.</p> <p>The value in PEAS is captured and held with the first interlock write done for Modify operations. The value of PEAS is retained between sampling events (that is, subsequent first interlock writes).</p> <p>0 Shadow/Test address space is disabled for program/erase and main address space enabled. 1 Shadow/Test address space is enabled for program/erase and main address space disabled.</p>
DONE	<p>Modify Operation Done</p> <p>DONE indicates if the flash module is performing a high voltage operation.</p> <p>DONE is set to 1 on termination of the flash module reset.</p> <p>DONE is cleared to 0 just after a 0-to-1 transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation.</p> <p>DONE is set to 1 at the end of program and erase high voltage sequences.</p> <p>DONE is set to 1 (within t_{PABT} or t_{EABT}, equal to P/E Abort Latency) after a 1-to-0 transition of EHV, which terminates a high voltage program/erase operation.</p> <p>DONE is set to 1 (within t_{ESUS}, time equal to Erase Suspend Latency) after a 0-to-1 transition of ESUS, which suspends an erase operation.</p> <p>0 Flash is executing a high voltage operation. 1 Flash is not executing a high voltage operation.</p>

Table 150. MCR field descriptions (continued)

Field	Description
PEG	<p>Program/Erase Good</p> <p>The PEG bit indicates the completion status of the last flash program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations.</p> <p>Aborting a program/erase high voltage operation causes PEG to be cleared to 0, indicating the sequence failed.</p> <p>PEG is set to 1 when the flash module is reset, unless a flash initialization error has been detected. The value of PEG is valid only when PGM = 1 and/or ERS = 1 and after DONE transitions from 0 to 1 due to a termination or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1-to-0 transition or EHV makes a 0-to-1 transition.</p> <p>The value in PEG is not valid after a 0-to-1 transition of DONE caused by ESUS being set to logic 1. If program or erase are attempted on blocks that are locked, the response is PEG = 1, indicating that the operation was successful, and the content of the block were properly protected from the program or erase operation.</p> <p>If a program operation tries to program at 1 bits that are at 0, the program operation is correctly executed on the new bits to be programmed at 0, but PEG is cleared, indicating that the requested operation has failed.</p> <p>In Array Integrity Check or Margin Mode, PEG is set to 1 when the operation is completed, regardless the occurrence of any error. The presence of errors can be detected only comparing checksum value stored in UMIRS[0:1].</p> <p>0 Program or erase operation failed; or program, erase, Array Integrity Check, or Margin Mode was terminated.</p> <p>1 Program or erase operation successful; or Array Integrity Check or Margin Mode completed successfully.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
PGM	<p>Program</p> <p>PGM sets up the flash module for a program operation.</p> <p>A 0-to-1 transition of PGM initiates a program sequence.</p> <p>A 1-to-0 transition of PGM ends the program sequence.</p> <p>PGM can be set only under User mode Read (ERS is low and UT0[AIE] is low). PGM can be cleared by the user only when EHV is low and DONE is high. PGM is cleared on reset.</p> <p>0 Flash is not executing a program sequence.</p> <p>1 Flash is executing a program sequence.</p>
PSUS	<p>Program Suspend</p> <p>Writing to this bit has no effect, but the written data can be read back.</p>
ERS	<p>Erase</p> <p>ERS sets up the flash module for an Erase operation.</p> <p>A 0-to-1 transition of ERS initiates an Erase sequence.</p> <p>A 1-to-0 transition of ERS ends the Erase sequence.</p> <p>ERS can be set only under User mode Read (PGM is low and UT0[AIE] is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset.</p> <p>0 Flash is not executing an Erase sequence.</p> <p>1 Flash is executing an Erase sequence.</p>

Table 150. MCR field descriptions (continued)

Field	Description
ESUS	<p>Erase Suspend</p> <p>ESUS indicates that the flash module is in Erase Suspend or in the process of entering a Suspend state. The flash module is in Erase Suspend when ESUS = 1 and DONE = 1.</p> <p>ESUS can be set high only when ERS = 1 and EHV = 1, and PGM = 0.</p> <p>A 0-to-1 transition of ESUS starts the sequence that sets DONE and places the flash in erase suspend. The flash module enters Suspend within t_{ESUS} of this transition.</p> <p>ESUS can be cleared only when DONE = 1 and EHV = 1, and PGM = 0.</p> <p>A 1-to-0 transition of ESUS with EHV = 1 starts the sequence that clears DONE and returns the Module to Erase.</p> <p>The flash module cannot exit Erase Suspend and clear DONE while EHV is low.</p> <p>ESUS is cleared on reset.</p> <p>0 Erase sequence is not suspended. 1 Erase sequence is suspended.</p>
EHV	<p>Enable High Voltage</p> <p>The EHV bit enables the flash module for a high voltage program/Erase operation. EHV is cleared on reset.</p> <p>EHV must be set after an interlock write to start a program/Erase sequence. EHV may be set under one of the following conditions:</p> <ul style="list-style-type: none"> • Erase (ERS = 1, ESUS = 0, UT0[AIE] = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, UT0[AIE] = 0) <p>In normal operation, a 1-to-0 transition of EHV with DONE high and ESUS low terminates the current program/Erase high voltage operation.</p> <p>When an operation is terminated, there is a 1-to-0 transition of EHV with DONE low and the eventual Suspend bit low. A termination causes the value of PEG to be cleared, indicating a failing program/Erase; address locations being operated on by the terminated operation contain indeterminate data after a termination. A suspended operation cannot be terminated. Terminating a high voltage operation leaves the flash module addresses in an indeterminate data state. This may be recovered by executing an Erase on the affected blocks.</p> <p>EHV may be written during Suspend. EHV must be high to exit Suspend. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low.</p> <p>0 Flash is not enabled to perform an high voltage operation. 1 Flash is enabled to perform an high voltage operation.</p>

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit by bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The flash module does not allow the user to write bits simultaneously which would put the device into an illegal state. This is implemented through a priority mechanism among the bits. [Table 151](#) shows the bit changing priorities.

Table 151. MCR bits set/clear priority levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS

If the user attempts to write two or more MCR bits simultaneously, only the bit with the lowest priority level is written.

18.3.6.2 Low/Mid Address Space Block Locking register (LML)

The Low/Mid Address Space Block Locking register provides a means to protect blocks from being modified. These bits, along with bits in the SLL register, determine if the block is locked from program or erase. An “OR” of LML and SLL determine the final lock status. Identical LML registers are provided in the code flash and the data flash blocks.

In the code flash module, the LML register has a related Non-Volatile Low/Mid Address Space Block Locking register (NVLML) located in TestFlash that contains the default reset value for LML. The NVLML register is read during the reset phase of the flash module and loaded into the LML. The reset value is 0x00XX_XXXX, initially determined by the NVLML value from test sector.

Address: Base + 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LME	0	0	0	0	0	0	0	0	0	0	TSLK	0	0	MLK1	MLK0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LLK	LLK	LLK	LLK	LLK	LLK	LLK	LLK
W									7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

Figure 146. Low/Mid Address Space Block Locking register (LML)

18.3.6.3 Non-Volatile Low/Mid Address Space Block Locking register (NVLML)

Address: Base + 0x40_3DE8 Delivery value:
0xFFFFFFFF

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LME	0	0	0	0	0	0	0	0	0	0	TSLK	0	0	MLK1	MLK0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LLK	LLK	LLK	LLK	LLK	LLK	LLK	LLK
W									7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

Figure 147. Non-Volatile Low/Mid Address Space Block Locking register (NVLML)

The NVLML register is a 64-bit register, the 32 most significant bits of which (bits 63:32) are “don’t care” bits that are eventually used to manage ECC codes. Identical NVLML registers are provided in the code flash and the data flash blocks.

Table 152. LML /NVLML field descriptions

Field	Description
LME ¹	<p>Low/Mid Address Space Block Enable</p> <p>This bit enables the Lock registers (TSLK and LLK[15:0]) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the LME bit is set to reflect the status of enabled, and is enabled until a reset operation occurs. For LME the password 0xA1A11111 must be written to the LML register.</p> <p>0 Low Address Locks are disabled: TSLK and LLK[15:0] cannot be written. 1 Low Address Locks are enabled: TSLK and LLK[15:0] can be written.</p>
TSLK	<p>Test/Shadow Address Space Block Lock</p> <p>This bit locks the block of Test and Shadow Address Space from program and Erase (Erase is any case forbidden for Test block).</p> <p>A value of 1 in the TSLK register signifies that the Test/Shadow block is locked for program and Erase. A value of 0 in the TSLK register signifies that the Test/Shadow block is available to receive program and Erase pulses.</p> <p>The TSLK register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation. Likewise, the TSLK register is not writable if a high voltage operation is suspended.</p> <p>Upon reset, information from the TestFlash block is loaded into the TSLK register. The TSLK bit may be written as a register. Reset will cause the bit to go back to its TestFlash block value. The default value of the TSLK bit (assuming erased fuses) would be locked.</p> <p>TSLK is not writable unless LME is high.</p> <p>0 Test/Shadow Address Space Block is unlocked and can be modified (if also SLL[STSLK] = 0). 1 Test/Shadow Address Space Block is locked and cannot be modified.</p>
MLK[1:0]	<p>Mid Address Space Block Lock 1-0</p> <p>These bits lock the blocks of Mid Address Space from program and Erase.</p> <p>MLK[1:0] are related to sectors B0F[7:6], respectively.</p> <p>A value of 1 in a bit of the MLK bitfield signifies that the corresponding block is locked for program and Erase. A value of 0 in a bit of the MLK bitfield signifies that the corresponding block is available to receive program and Erase pulses.</p> <p>The MLK bitfield is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation. Likewise, the MLK bitfield is not writable if a high voltage operation is suspended. Upon reset, information from the TestFlash block is loaded into the MLK bitfields. The MLK bits may be written as a register. Reset causes the bits to revert to their TestFlash block value. The default value of the MLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the MLK bits default to locked, and are not writable. The reset value will always be 1 (independent of the TestFlash block), and register writes will have no effect.</p> <p>MLK is not writable unless LME is high.</p> <p>0 Mid Address Space Block is unlocked and can be modified (if also SLL[SMLK] = 0). 1 Mid Address Space Block is locked and cannot be modified.</p>

Table 152. LML /NVLML field descriptions (continued)

Field	Description
LLK[7:0]	<p>LLK7-0: Low address space block Lock 7-0 (Read/Write) These bits are used to lock the blocks of Low Address Space from Program and Erase. LLK7-0 are related to sectors B0F7-0, respectively. A value of 1 in a bit of the LLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the LLK register signifies that the corresponding block is available to receive Program and Erase pulses. The LLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the LLK register is not writable if a high voltage operation is suspended. Upon reset, information from the TestFlash block is loaded into the LLK registers. The LLK bits may be written as a register. Reset will cause the bits to go back to their TestFlash block value. The default value of the LLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the LLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the TestFlash block), and register writes will have no effect. LLK is not writable unless LME is high. 0: Low Address Space Block is unlocked and can be modified (if also SLL.SLK=0). 1: Low Address Space Block is locked and cannot be modified.</p>

¹ This field is present only in LML

18.3.6.4 High address space Block Locking register (HBL)

The High Address Space Block Locking (HBL) register provides a means to protect blocks from being modified.

The HBL register has a related Non Volatile High Address Space Block Locking register located in TestFlash that contains the default reset value for HBL: the NVHBL register is read during the reset phase of the Flash Module and loaded into the HBL.

The NVHBL register is a 64 bit register, the 32 most significant bits of which (bits 63-32) are don't care and eventually used to manage ECC codes.

Address: Base + 0x0008 Access: User read-only

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

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

Figure 148. High address space Block Locking register (HBL)

18.3.6.5 Non Volatile High address space Block Locking register (NVHBL)

Address: Base + 0x403DF0

 Delivery value:
0xFFFFFFFF

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

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

Figure 149. Non Volatile High address space Block Locking register (NVHBL)

Table 153. HBL/NVHBL field descriptions

Field	Description
HBE	High address space Block Enable (Read Only) This bit is used to enable the Lock registers (HLK) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the HBE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For HBE the password 0xB2B22222 must be written to the HBL register. 0: High Address Locks are disabled: none cannot be written. 1: High Address Locks are enabled: none can be written.

18.3.6.6 Secondary Low/Mid Address Space Block Locking register (SLL)

The Secondary Low/Mid Address Space Block Locking register provides an alternative means to protect blocks from being modified. These bits, along with bits in the LML register, determine if the block is locked from program or Erase. An “OR” of LML and SLL determine the final lock status. Identical SLL registers are provided in the code flash and the data flash blocks.

In the code flash module, the SLL register has a related Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL) located in TestFlash that contains the default reset value for SLL. The reset value is 0x00XX_XXXX, initially determined by NVSLL.

The NVSLL register is read during the reset phase of the flash module and loaded into the SLL.

Address: Base + 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLE	0	0	0	0	0	0	0	0	0	0	STS	0	0	SMK	SMK
W												LK			1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	SLK	SLK	SLK	SLK	SLK	SLK	SLK	SLK
W									7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

Figure 150. Secondary Low/mid address space block Locking reg (SLL)

18.3.6.7 Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL)

The NVSLL register is a 64-bit register, the 32 most significant bits of which (bits 63:32) are “don’t care” bits that are eventually used to manage ECC codes. Identical NVSLL registers are provided in the code flash and the data flash blocks.

Address: Base + 0x40_3DF8 Delivery value:
0xFFFFFFFF

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLE	0	0	0	0	0	0	0	0	0	0	STS	0	0	SMK	SMK
W												LK			1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	SLK	SLK	SLK	SLK	SLK	SLK	SLK	SLK
W									7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

Figure 151. Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL)

Table 154. SLL and NVSLL field descriptions

Field	Description
SLE ¹	<p>Secondary Low/Mid Address Space Block Enable</p> <p>This bit enables the Lock registers (STSLK and SLK[15:0]) to be set or cleared by registers writes. This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the SLE bit is set to reflect the status of enabled, and is enabled until a reset operation occurs. For SLE the password 0xC3C3_3333 must be written to the SLL register.</p> <p>0 Secondary Low/Mid Address Locks are disabled: STSLK and SLK[15:0] cannot be written.</p> <p>1 Secondary Low/Mid Address Locks are enabled: STSLK and SLK[15:0] can be written.</p>

Table 154. SLL and NVSLL field descriptions (continued)

Field	Description
STSLK	<p>Secondary Test/Shadow address space block Lock</p> <p>This bit is used as an alternate means to lock the block of Test and Shadow Address Space from program and Erase (Erase is any case forbidden for Test block).</p> <p>A value of 1 in the STSLK bitfield signifies that the Test/Shadow block is locked for program and Erase.</p> <p>A value of 0 in the STSLK register signifies that the Test/Shadow block is available to receive program and Erase pulses.</p> <p>The STSLK register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation. Likewise, the STSLK register is not writable if a high voltage operation is suspended.</p> <p>Upon reset, information from the TestFlash block is loaded into the STSLK register. The STSLK bit may be written as a register. Reset will cause the bit to go back to its TestFlash block value. The default value of the STSLK bit (assuming erased fuses) would be locked.</p> <p>STSLK is not writable unless SLE is high.</p> <p>0 Test/Shadow Address Space Block is unlocked and can be modified (if also LML[TSLK] = 0).</p> <p>1 Test/Shadow Address Space Block is locked and cannot be modified.</p>
SMK[1:0]	<p>Secondary Mid Address Space Block Lock 1–0</p> <p>These bits are used as an alternate means to lock the blocks of Mid Address Space from program and Erase.</p> <p>SMK[1:0] are related to sectors B0F[7:6], respectively.</p> <p>A value of 1 in a bit of the SMK register signifies that the corresponding block is locked for program and Erase.</p> <p>A value of 0 in a bit of the SMK register signifies that the corresponding block is available to receive program and Erase pulses.</p> <p>The SMK register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation. Likewise, the SMK register is not writable if a high voltage operation is suspended.</p> <p>Upon reset, information from the TestFlash block is loaded into the SMK registers. The SMK bits may be written as a register. Reset will cause the bits to go back to their TestFlash block value. The default value of the SMK bits (assuming erased fuses) would be locked.</p> <p>In the event that blocks are not present (due to configuration or total memory size), the SMK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the TestFlash block), and register writes have no effect. SMK is not writable unless SLE is high.</p> <p>0 Mid Address Space Block is unlocked and can be modified (if also LML[MLK] = 0).</p> <p>1 Mid Address Space Block is locked and cannot be modified.</p>

Table 154. SLL and NVSLL field descriptions (continued)

Field	Description
SLK[7:0]	<p>SLK7-0: <i>Secondary Low address space block lock 7-0 (Read/Write)</i></p> <p>These bits are used as an alternate means to lock the blocks of Low Address Space from Program and Erase.</p> <p>SLK7-0 are related to sectors B0F7-0, respectively.</p> <p>A value of 1 in a bit of the SLK register signifies that the corresponding block is locked for Program and Erase.</p> <p>A value of 0 in a bit of the SLK register signifies that the corresponding block is available to receive Program and Erase pulses.</p> <p>The SLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the SLK register is not writable if a high voltage operation is suspended.</p> <p>Upon reset, information from the TestFlash block is loaded into the SLK registers. The SLK bits may be written as a register. Reset will cause the bits to go back to their TestFlash block value. The default value of the SLK bits (assuming erased fuses) would be locked.</p> <p>In the event that blocks are not present (due to configuration or total memory size), the SLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the TestFlash block), and register writes will have no effect.</p> <p>SLK is not writable unless SLE is high.</p> <p>0: Low Address Space Block is unlocked and can be modified (if also LML.LLK=0).</p> <p>1: Low Address Space Block is locked and cannot be modified.</p>

¹ This field is present only in SLL

18.3.6.8 Low/Mid Address Space Block Select register (LMS)

The Low/Mid Address Space Block Select register provides a means to select blocks to be operated on during erase. Identical LMS registers are provided in the code flash and the data flash blocks.

Address: Base + 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSL	MSL
W															1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	LSL	LSL	LSL	LSL	LSL	LSL	LSL	LSL
W									7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 152. Low/Mid Address Space Block Select register (LMS)

Table 155. LMS field descriptions

Field	Description
MSL[1:0]	<p>Mid Address Space Block Select 1–0</p> <p>A value of 1 in the select register signifies that the block is selected for erase.</p> <p>A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected.</p> <p>MSL[1:0] are related to sectors B0F[7:6], respectively.</p> <p>The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended.</p> <p>In the event that blocks are not present (due to configuration or total memory size), the corresponding MSL bits default to unselected, and are not writable. The reset value will always be 0, and register writes have no effect.</p> <p>0 Mid Address Space Block is unselected for Erase. 1 Mid Address Space Block is selected for Erase.</p>
LSL[7:0]	<p>LSL7-0: Low address space block SeLect 7-0 (Read/Write)</p> <p>A value of 1 in the select register signifies that the block is selected for erase.</p> <p>A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected.</p> <p>LSL7-0 are related to sectors B0F7-0, respectively.</p> <p>The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended.</p> <p>In the event that blocks are not present (due to configuration or total memory size), the corresponding LSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect.</p> <p>0: Low Address Space Block is unselected for Erase. 1: Low Address Space Block is selected for Erase.</p>

18.3.6.9 High address space Block Select register (HBS)

The High Address Space Block Select register provides a means to select blocks to be operated on during erase.

Address: Base + 0x0014 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 153. High address space Block Select register (HBS)

18.3.6.10 Address Register (ADR)

The Address Register provides the first failing address in the event module failures (ECC, RWW, or FPEC) or the first address at which a ECC single error correction occurs.

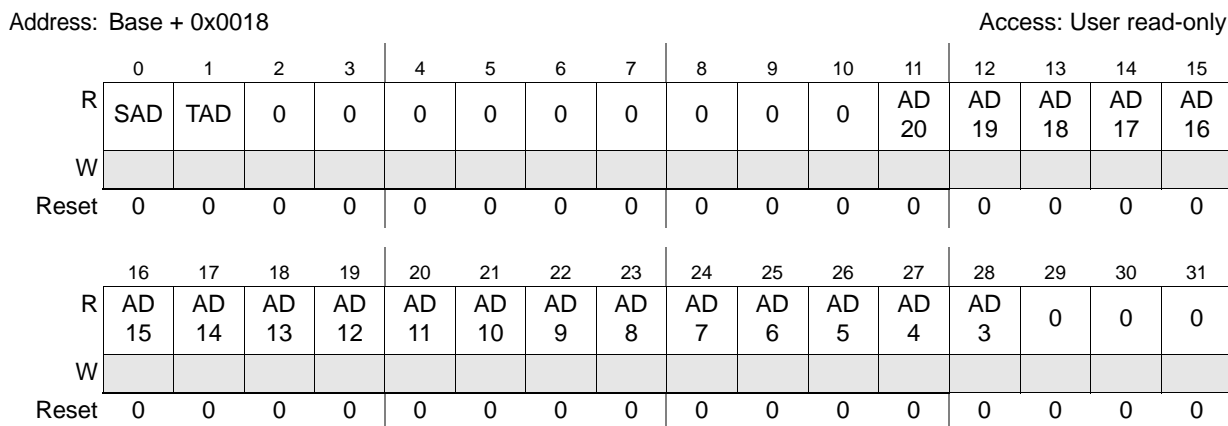


Figure 154. Address Register (ADR)

Table 156. ADR field descriptions

Field	Description
SAD	SAD: Shadow Address (Read Only) When this bit is high, the address indicated by AD20-3 belongs to the Shadow Sector.
TAD	TAD: Test Address (Read Only) When this bit is high, the address indicated by AD20-3 belongs to the Test Sector.
AD[20:3]	<p>Address 20–3</p> <p>ADR provides the first failing address in the event of ECC error (MCR[EER] set) or the first failing address in the event of RWW error (MCR[RWE] set), or the address of a failure that may have occurred in a FPEC operation (MCR[PEG] cleared). ADR also provides the first address at which a ECC single error correction occurs (MCR[EDC] set).</p> <p>The ECC double error detection takes the highest priority, followed by the RWW error, the FPEC error, and the ECC single error correction. When accessed ADR will provide the address related to the first event occurred with the highest priority. The priorities between these four possible events is summarized in Table 157.</p> <p>This address is always a double word address that selects 64 bits.</p> <p>In case of a simultaneous ECC double error detection on both double words of the same page, bit AD3 will output 0. The same is valid for a simultaneous ECC single error correction on both double words of the same page.</p> <p>In User mode, ADR is read only.</p>

Table 157. ADR content: priority list

Priority level	Error flag	ADR content
1	MCR[EER] = 1	Address of first ECC Double Error
2	MCR[RWE] = 1	Address of first RWW Error
3	MCR[PEG] = 0	Address of first FPEC Error
4	MCR[EDC] = 1	Address of first ECC Single Error Correction

18.3.6.11 Bus Interface Unit 0 register (BIU0)

Address offset: 0x0001C

Reset value: 0xXXXXX_XXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI031	BI030	BI029	BI028	BI027	BI026	BI025	BI024	BI023	BI022	BI021	BI020	BI019	BI018	BI017	BI016
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI015	BI014	BI013	BI012	BI011	BI010	BI009	BI008	BI007	BI006	BI005	BI004	BI003	BI002	BI001	BI000
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 155. Bus Interface Unit 0 register (BIU0)

The Bus Interface Unit 0 Register provides a means for BIU specific information or BIU configuration information to be stored. Please refer to [Section 18.2.4.2.1, “Platform Flash Configuration Register 0 \(PFCR0\)”](#) for more information about register description.

Table 158. BIU0 field descriptions

Field	Description
BI0	BI0[31:00]: <i>Bus Interface unit 0 31-00</i> (Read/Write) The writability of the bits in this register can be locked.

18.3.6.12 Bus Interface Unit 1 register (BIU1)

Address offset: 0x00020

Reset value: 0xXXXXX_XXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI131	BI130	BI129	BI128	BI127	BI126	BI125	BI124	BI123	BI122	BI121	BI120	BI119	BI118	BI117	BI116
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI115	BI114	BI113	BI112	BI111	BI110	BI109	BI108	BI107	BI106	BI105	BI104	BI103	BI102	BI101	BI100
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 156. Bus Interface Unit 1 register (BIU1)

The Bus Interface Unit 1 Register provides a means for BIU specific information or BIU configuration information to be stored. Please refer to [Section 18.2.4.2.2, “Platform Flash Configuration Register 1 \(PFCR1\)”](#) for more information about register description.

Table 159. BIU1 field descriptions

Field	Description
BI1	BI1[31:00]: <i>Bus Interface unit 1 31-00</i> (Read/Write) The writability of the bits in this register can be locked.

18.3.6.13 Bus Interface Unit 2 register (BIU2)

Address offset: 0x00024

Reset value: 0xXXXX XXXX

Please refer to [Section 18.2.4.2.3, “Platform Flash Access Protection Register \(PFAPR\)”](#) to see register description.

18.3.6.13.1 Non-volatile Bus Interface Unit 2 register (NVBIU2)

Address offset: 0x003E00¹

Delivery value: 0xXXXXX_XXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI231	BI230	BI229	BI228	BI227	BI226	BI225	BI224	BI223	BI222	BI221	BI220	BI219	BI218	BI217	BI216
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI215	BI214	BI213	BI212	BI211	BI210	BI209	BI208	BI207	BI206	BI205	BI204	BI203	BI202	BI201	BI200
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

Figure 157. Bus Interface Unit 2 register (BIU2)

¹ See device memory map table for base address information of shadow flash.

The Bus Interface Unit 2 Register provides a means for BIU specific information or BIU configuration information to be stored. Please refer to [Section 18.2.4.2.3, “Platform Flash Access Protection Register \(PFAPR\)”](#) for more information about register description.

The BIU2 register has a related Non-volatile Bus Interface Unit 2 register located in the Shadow Sector that contains the default reset value for BIU2. During the reset phase of the Flash module, the NVBIU2 register content is read and loaded into the BIU2.

The NVBIU2 register is a 64-bit register, of which the 32 most significant bits 63:32 are ‘don’t care’ and eventually used to manage ECC codes.

Table 160. BIU2 field descriptions

Field	Description
	<p>BI2[31:00]: Bus Interface unit 2 31-00 (Read/Write) The BI2[31:00] generic registers are reset based on the information stored in NVBIU2. The writability of the bits in this register can be locked.</p>

18.3.6.14 Non Volatile Bus Interface Unit 3 register (NVBIU3)

Address offset: 0x003E08¹

Delivery value: 0xXXXXX_XXXX

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BI331	BI330	BI329	BI328	BI327	BI326	BI325	BI324	BI323	BI322	BI321	BI320	BI319	BI318	BI317	BI316
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BI315	BI314	BI313	BI312	BI311	BI310	BI309	BI308	BI307	BI306	BI305	BI304	BI303	BI302	BI301	BI300
rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X	rw/X

¹ See device memory map table for base address information of shadow flash.

The Bus Interface Unit 3 Register provides a means for BIU specific information or BIU configuration information to be stored.

The BIU3 register has a related Non-Volatile Bus Interface Unit 3 register located in the Shadow Sector that contains the default reset value for BIU3. the NVBIU3 register is read during the reset phase of the Flash Module and loaded into the BIU3.

The NVBIU3 register is a 64-bit register, the 32 most significant bits of which (bits 63:32) are ‘don’t care’ and eventually used to manage ECC codes.

Table 161. BIU3 field descriptions

Field	Description
	<p>BI331-00: <i>Bus Interface unit 3 31-00 (Read/Write)</i> The BI331-00 generic registers are reset based on the information stored in NVBIU3. The writability of the bits in this register can be locked. The use of this bus is SoC specific.</p>

18.3.6.15 User Test 0 register (UT0)

The User Test feature gives the user of the flash module the ability to perform test features on the flash. The User Test 0 register allows controlling the way in which the flash content check is done.

The UT0[MRE], UT0[MRV], UT0[AIS], UT0[EIE], and DSI[7:0] bits are not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x003C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBC	0	0	0	0	0	0	DSI7	DSI6	DSI5	DSI4	DSI3	DSI2	DSI1	DSI0
W		E														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	X	MRE	MRV	EIE	AIS	AIE	AID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 158. User Test 0 register (UT0)

Table 162. UT0 field descriptions

Field	Description
UTE	<p>User Test Enable</p> <p>This status bit indicates when User Test is enabled. All bits in UT0–2 and UMISR0–4 are locked when this bit is 0.</p> <p>This bit is not writeable to a 1, but may be cleared. The reset value is 0.</p> <p>The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write.</p> <p>For UTE the password 0xF9F9_9999 must be written to the UT0 register.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
DSI7-0	<p>Data Syndrome Input 7–0</p> <p>These bits represent the input of Syndrome bits of ECC logic used in the ECC Logic Check. The DSI7–0 bits correspond to the 8 syndrome bits on a double word.</p> <p>These bits are not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 The syndrome bit is forced at 0.</p> <p>1 The syndrome bit is forced at 1.</p>
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
	<p><i>Reserved (Read/Write)</i></p> <p>This bit can be written and its value can be read back, but there is no function associated.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p>
MRE	<p>Margin Read Enable</p> <p>MRE enables margin reads to be done. This bit, combined with MRV, enables regular user mode reads to be replaced by margin reads.</p> <p>Margin reads are only active during Array Integrity Checks; Normal user reads are not affected by MRE.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 Margin reads are disabled. All reads are User mode reads.</p> <p>1 Margin reads are enabled.</p>

Table 162. UT0 field descriptions (continued)

Field	Description
MRV	<p>Margin Read Value</p> <p>If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV = 1) or to a programmed level (MRV = 0).</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 Zero's (programmed) margin reads are requested (if MRE = 1).</p> <p>1 One's (erased) margin reads are requested (if MRE = 1).</p>
EIE	<p>ECC data Input Enable</p> <p>EIE enables the ECC Logic Check operation to be done.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 ECC Logic Check is disabled.</p> <p>1 ECC Logic Check is enabled.</p>
AIS	<p>Array Integrity Sequence</p> <p>AIS determines the address sequence to be used during array integrity checks or Margin Mode. The default sequence (AIS = 0) is meant to replicate sequences normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary.</p> <p>The alternative sequence (AIS = 1) is just logically sequential.</p> <p>It should be noted that the time to run a sequential sequence is significantly shorter than the time to run the proprietary sequence.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect. In Margin Mode only the linear sequence (AIS = 1) is allowed, while the proprietary sequence (AIS = 0) is forbidden.</p> <p>0 Array Integrity sequence is a proprietary sequence.</p> <p>1 Array Integrity or Margin Mode sequence is sequential.</p>
AIE	<p>Array Integrity Enable</p> <p>AIE set to 1 starts the Array Integrity Check done on all selected and unlocked blocks. The pattern is selected by AIS, and the MISR (UMISR0–4) can be checked after the operation is complete, to determine if a correct signature is obtained.</p> <p>AIE can be set only if MCR[ERS], MCR[PGM], and MCR[EHV] are all low.</p> <p>0 Array Integrity Checks are disabled.</p> <p>1 Array Integrity Checks are enabled.</p>
AID	<p>Array Integrity Done</p> <p>AID is cleared upon an Array Integrity Check being enabled (to signify the operation is on-going). Once completed, AID is set to indicate that the Array Integrity Check is complete. At this time, the MISR (UMISR0–4) can be checked.</p> <p>0 Array Integrity Check is on-going.</p> <p>1 Array Integrity Check is done.</p>

18.3.6.16 User Test 1 register (UT1)

The User Test 1 register allows to enable the checks on the ECC logic related to the 32 LSB of the Double Word.

The User Test 1 register is not accessible whenever MCR[**DONE**] or UT0[**AID**] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 159. User Test 1 register (UT1)

Table 163. UT1 field descriptions

Field	Description
DAI[31:0]	Data Array Input 31–0 These bits represent the input of the even word of ECC logic used in the ECC Logic Check. The DAI[31:0] bits correspond to the 32 array bits representing Word 0 within the double word. 0 The array bit is forced at 0. 1 The array bit is forced at 1.

18.3.6.17 User Test 2 register (UT2)

The User Test 2 register allows to enable the checks on the ECC logic related to the 32 MSB of the Double Word.

The User Test 2 register is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0044 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 160. User Test 2 register (UT2)

Table 164. UT2 field descriptions

Field	Description
DAI[63:32]	Data Array Input [63:32] These bits represent the input of the odd word of ECC logic used in the ECC Logic Check. The DAI[63:32] bits correspond to the 32 array bits representing Word 1 within the double word. 0 The array bit is forced at 0. 1 The array bit is forced at 1.

18.3.6.18 User Multiple Input Signature Register 0 (UMISR0)

The Multiple Input Signature Register 0 (UMISR0) provides a mean to evaluate the array integrity. UMISR0 represents the bits 31:0 of the whole 144-bit word (2 double words including ECC).

UMISR0 is not accessible whenever MCR[*DONE*] or UT0[*AID*] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0048

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	031	030	029	028	027	026	025	024	023	022	021	020	019	018	017	016
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	015	014	013	012	011	010	009	008	007	006	005	004	003	002	001	000
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 161. User Multiple Input Signature Register 0 (UMISR0)
Table 165. UMISR0 field descriptions

Field	Description
MS[031:000]	Multiple input Signature 031–000 These bits represent the MISR value obtained by accumulating the bits 31:0 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR0 register.

18.3.6.19 User Multiple Input Signature Register 1 (UMISR1)

The Multiple Input Signature Register 1 (UMISR1) provides a means to evaluate the array integrity. UMISR1 represents bits 63:32 of the whole 144-bit word (2 double words including ECC).

UMISR1 is not accessible whenever MCR[*DONE*] or UT0[*AID*] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x004C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	063	062	061	060	059	058	057	056	055	054	053	052	051	050	049	048
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	047	046	045	044	043	042	041	040	039	038	037	036	035	034	033	032
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 162. User Multiple Input Signature Register 1 (UMISR1)

Table 166. UMISR1 field descriptions

Field	Description
MS[063:032]	Multiple input Signature 063–032 These bits represent the MISR value obtained accumulating the bits 63:32 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR1 register.

18.3.6.20 User Multiple Input Signature Register 2 (UMISR2)

The Multiple Input Signature Register (UMISR2) provides a mean to evaluate the array integrity. UMISR2 represents the bits 95-64 of the whole 144-bit word (2 double words including ECC).

UMISR2 is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0050 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	095	094	093	092	091	090	089	088	087	086	085	084	083	082	081	080
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	079	078	077	076	075	074	073	072	071	070	069	068	067	066	065	064
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 163. User Multiple Input Signature Register 2 (UMISR2)

Table 167. UMISR2 field descriptions

Field	Description
MS[095:064]	Multiple input Signature 095–064 These bits represent the MISR value obtained by accumulating the bits 95:64 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR2 register.

18.3.6.21 User Multiple Input Signature Register 3 (UMISR3)

The Multiple Input Signature Register 3 (UMISR3) provides a means to evaluate the array integrity. UMISR3 represents bits 127:96 of the whole 144-bit word (2 double words including ECC).

UMISR3 is not accessible whenever MCR[**DONE**] or UT0[**AID**] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0054 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	111	110	109	108	107	106	105	104	103	102	101	100	099	098	097	096
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 164. User Multiple Input Signature Register 3 (UMISR3)

Table 168. UMISR3 field descriptions

Field	Description
MS[127:096]	Multiple Input Signature 127–096 These bits represent the MISR value obtained accumulating bits 127:96 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR3 register.

18.3.6.22 User Multiple Input Signature Register 4 (UMISR4)

The Multiple Input Signature Register 4 (UMISR4) provides a means to evaluate the array integrity. The UMISR4 represents the ECC bits of the whole 144-bit word (2 double words including ECC). Bits 8:15 are ECC bits for the odd double word and bits 24:31 are the ECC bits for the even double word. Bits 4:5 and 20:21 of UMISR4 are the double and single ECC error detection for odd and even double words, respectively.

UMISR4 is not accessible whenever MCR[**DONE**] or UT0[**AID**] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0058 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 165. User Multiple Input Signature Register 4 (UMISR4)

Table 169. UMISR4 field descriptions

Field	Description
MS[159:128]	Multiple Input Signature 159:128 These bits represent the MISR value obtained accumulating: <ul style="list-style-type: none"> • MS[135:128]—8 ECC bits for the even double word • MS138—Single ECC error detection for even double word • MS139—Double ECC error detection for even double word • MS[151:144]—8 ECC bits for the odd double word • MS154—Single ECC error detection for odd double word • MS155—Double ECC error detection for odd double word The MS can be seeded to any value by writing the UMISR4 register.

18.3.6.23 Non-Volatile Private Censorship Password 0 register (NVPWD0)

The Non-Volatile Private Censorship Password 0 register (NVPWD0) contains the 32 LSB of the password used to validate the Censorship information contained in NVSCI0–1 registers.

NOTE

This register is not implemented on the data flash block.

Address: 0x20_3DD8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 166. Non-Volatile private Censorship Password 0 register (NVPWD0)

Table 170. NVPWD0 field descriptions

Field	Description
PWD[31:0]	Password 31–0 The PWD[31:0] bits represent the 32 LSB of the private censorship password.

18.3.6.24 Non-Volatile Private Censorship Password 1 register (NVPWD1)

The Non-Volatile Private Censorship Password 1 Register (NVPWD1) contains the 32 MSB of the password used to validate the Censorship information contained in NVSCI0–1 registers.

NOTE

This register is not implemented on the data flash block.

Address: 0x20_3DDC

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD
W	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD	PWD
W	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 167. Non-Volatile Private Censorship Password 1 register (NVPWD1)
Table 171. NVPWD1 field descriptions

Field	Description
PWD63–32	<i>PassWord</i> 63–32 The PWD63–32 registers represent the 32 MSB of the Private Censorship Password.

18.3.6.25 Non-Volatile System Censoring Information 0 register (NVSCI0)

The Non-Volatile System Censoring Information 0 register (NVSCI0) stores the 32 LSB of the Censorship Control Word of the device.

NVSCI0 is a non-volatile register located in Shadow sector. It is read during the reset phase of the flash module and the protection mechanisms are activated consequently.

The parts are delivered uncensored to the user. Delivery value: 0x55AA_55AA.

NOTE

This register is not implemented on the data flash block.

Address: 0x20_3DE0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0

Figure 168. Non-Volatile System Censoring Information 0 register (NVSCI0)

Table 172. NVSCI0 field descriptions

Field	Description
SC[15:0]	Serial Censorship control word 15–0 These bits represent the 16 LSB of the Serial Censorship Control Word (SCCW). If SC[15:0] = 0x55AA and NVSCI1 = NVSCI0, the Public Access is disabled. If SC[15:0] ≠ 0x55AA or NVSCI1 ≠ NVSCI0, the Public Access is enabled.
CW[15:0]	Censorship control Word 15–0 These bits represent the 16 LSB of the Censorship Control Word (CCW). If CW[15:0] = 0x55AA and NVSCI1 = NVSCI0, the Censored mode is disabled. If CW[15:0] ≠ 0x55AA or NVSCI1 ≠ NVSCI0, the Censored mode is enabled.

18.3.6.26 Non-Volatile System Censoring Information 1 register (NVSCI1)

The Non-Volatile System Censoring Information 1 register (NVSCI1) stores the 32 MSB of the Censorship Control Word of the device.

NVSCI1 is a non-volatile register located in Shadow sector. It is read during the reset phase of the flash module and the protection mechanisms are activated consequently.

The parts are delivered uncensored to the user. Delivery value: 0x55AA_55AA.

NOTE

This register is not implemented on the data flash block.

Address: 0x20_3DE4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC	SC
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW	CW
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0

Figure 169. Non-Volatile System Censoring Information 1 register (NVSCI1)

Table 173. NVSCI1 field descriptions

Field	Description
SC[32:16]	Serial Censorship control word 32–16 These bits represent the 16 MSB of the Serial Censorship Control Word (SCCW). If SC[32:16] = 0x55AA and NVSCI1 = NVSCI0, the Public Access is disabled. If SC[32:16] ≠ 0x55AA or NVSCI1 ≠ NVSCI0, the Public Access is enabled.
CW[32:16]	Censorship control Word 32–16 These bits represent the 16 MSB of the Censorship Control Word (CCW). CW[32:16] = 0x55AA and NVSCI1 = NVSCI0, the Censored mode is disabled. CW[32:16] ≠ 0x55AA or NVSCI1 ≠ NVSCI0, the Censored mode is enabled.

18.3.6.27 Non-Volatile User Options register (NVUSRO)

The Non-Volatile User Options Register (NVUSRO) contains configuration information for the user application.

NVUSRO is a 64-bit register, the 32 most significant bits of which (bits 63:32) are “don’t care” bits that are eventually used to manage ECC codes.

NOTE

This register is not implemented on the data flash block.

Address: 0x20_3E18 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UO31	UO30	UO29	UO28	UO27	UO26	UO25	UO24	UO23	UO22	UO21	UO20	UO19	UO18	UO17	UO16
W																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UO15	UO14	UO13	UO12	UO11	UO10	UO9	UO8	UO7	UO6	UO5	UO4	UO3	0	OSCI LLAT OR_ MARG IN	WAT CH DOG _EN
W																
Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 170. Non-Volatile User Options register (NVUSRO)

The Non-Volatile User Options Register (NVUSRO) contains configuration information for the user application.

NVUSRO is a 64-bit register, the 32 most significant bits of which (bits 63:32) are “don’t care” bits that are eventually used to manage ECC codes.

Table 174. NVUSRO field descriptions

Field	Description
UO	User Options 31–3 The UO[31:3] bits are reset based on the information stored in NVUSRO.

Table 174. NVUSRO field descriptions (continued)

Field	Description
OSCILLATOR_MARGIN	Oscillator Margin 0 Low consumption configuration (4 MHz/8 MHz). 1 High margin configuration (4/16 MHz). Default manufacturing value before flash initialization is '1'
WATCHDOG_EN	Watchdog Enable 0 Disable after reset. 1 Enable after reset. Default manufacturing value before flash initialization is '1'

18.3.7 Programming Considerations

NOTE

Like all flash memory, before an arbitrary value can be written to a memory location in flash on these devices, the block containing that address must be erased (all values set to “1”). The electrical characteristics of flash memory allow write operations to only transition individual bits from “1” to “0”, and to perform erase operations only at the block-level.

18.3.7.1 Modify Operations

All the modify operations of the flash modules are managed through the flash array control registers. All blocks of each flash array module belong to the same partition (bank), therefore when a modify operation is active on some blocks no read access is possible on any other block within the same array module.

During a flash modify operation any attempt to read any flash location within the same module will output invalid data and bit RWE of MCR will be automatically set. This means that the flash module is not fetchable when a modify operation is active within the same array module: the modify operation commands must be executed from another array.

If during a modify operation a reset occurs, the operation is suddenly interrupted and the array is reset to Read Mode. The data integrity of the flash section where the modify operation has been aborted is not guaranteed: the interrupted flash modify operation must be repeated.

In general each modify operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR (PGM or ERS) or UT0 (MRE or EIE).
2. The second step is the definition of the operands: the Address and the Data for programming or the blocks for erase or factory margin read.
3. The third instruction is used to start the modify operation, by setting EHV in MCR or AIE in the UT0 register.

Once selected, but not yet started, one operation can be canceled by resetting the operation selection bit.

A summary of the available flash modify operations are shown in [Table 175](#).

Table 175. Flash Modify Operations

Operation	Select bit	Operands	Start bit
Double Word Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Block Erase	MCR.ERS	LMS, HBS	MCR.EHV
Array Integrity Check ¹	None	LMS, HBS	UT0.AIE
Factory Margin Read ¹	UT0.MRE	UT0.MRV + LMS, HBS	UT0.AIE
ECC Logic Check ¹	UT0.EIE	UT0.DSI, UT1, UT2	UT0.AIE

¹ This operation is executed from User Test Mode. See [Section 18.3.7.1.4, "User Test Mode"](#), for details.

In general each modify operation is completed through a sequence of 4 steps:

1. Wait for operation completion: wait for bit MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UMISR0-4 with expected value).
3. Switch-Off flash controller by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE/EIE).

If a modify operation is on-going in an array then it is forbidden to start any other modify operation in the other arrays on the device.

In the following sections all modify operations are described and some examples of the sequences needed to activate them are presented.

18.3.7.1.1 Double Word Program

A flash program sequence operates on any double word within the flash. Up to 2 words within the double word may be altered in a single program operation. During a program operation, ECC bits are programmed. ECC is handled on a 64 bit boundary. Thus, if only 1 word in any given 64 bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed since ECC calculation has already completed for that 64 bit segment. Attempts to program the adjoining word will result in an operation failure. It is recommended that all programming operations be of 64 bits. The programming operation should completely fill selected ECC segments within the double word.

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

Addresses in locked/disabled blocks cannot be programmed.

You can program the values in any or all of 2 words, of a double word, with a single program sequence.

Double word-bound words have addresses which differ only in address bit 2.

The Program operation consists of the following sequence of events:

1. Change the value in the MCR.PGM bit from 0 to 1.
2. Ensure the block that contains the address to be programmed is unlocked.
 - Write the first address to be programmed with the program data.
 - The flash module latches address bits (22:3) at this time.

- The flash module latches data written as well.
 - This write is referred to as a program data interlock write. An interlock write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than one word is to be programmed, write the additional address in the double word with data to be programmed. This is referred to as a program data write.
The flash modules ignore address bits (22:3) for program data writes.
The eventual unwritten data word default to 0xFFFFFFFF.
 4. Write a logic 1 to the MCR.EHV bit to start the internal program sequence or skip to step 9 to terminate.
 5. Wait until the MCR.DONE bit goes high.
 6. Confirm MCR.PEG=1.
 7. Write a logic 0 to the MCR.EHV bit.
 8. If more addresses are to be programmed, return to step 2.
 9. Write a logic 0 to the MCR.PGM bit to terminate the program operation.

A program may be initiated with the 0 to 1 transition of the MCR.PGM bit or by clearing the MCR.EHV bit at the end of a previous program.

The first write after a program is initiated determines the page address to be programmed. This first write is referred to as an interlock write. The interlock write determines if the shadow or normal array space will be programmed by causing MCR.PEAS to be set/cleared.

An interlock write must be performed before setting MCR.EHV. An application may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

While MCR.DONE is low and MCR.EHV is high, an application may clear EHV, resulting in a program abort.

A program abort forces the Module to step 8 of the program sequence.

An aborted program will result in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the abort will contain indeterminate data. This may be recovered by repeating the same program instruction with the same data or executing an erase of the affected blocks.

Example 18-1. Double Word Program of data 0x55AA55AA at address 0x00AAA8 and data 0xAA55AA55 at address 0x00AAAC.

```

MCR = 0x00000010; /* Set PGM in MCR: Select PGM Operation */
(0x00AAA8) = 0x55AA55AA; /* Latch Address and 32 LSB data */
(0x00AAAC) = 0xAA55AA55; /* Latch 32 MSB data */
MCR = 0x00000011; /* Set EHV in MCR: Operation Start */
do /* Loop to wait for DONE=1 */
{ tmp = MCR; /* Read MCR */
} while ( !(tmp & 0x00000400) );
status = MCR & 0x00000200; /* Check PEG flag */
MCR = 0x00000010; /* Reset EHV in MCR: Operation End */
MCR = 0x00000000; /* Reset PGM in MCR: Deselect Operation */

```


18.3.7.1.2 Block Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the low, mid or high address space, or the shadow block (if available).

- The erase sequence is fully automated within the flash. an application only needs to select the blocks to be erased and initiate the erase sequence.
- Locked/disabled blocks cannot be erased.
- If multiple blocks are selected for erase during an erase sequence, no specific operation order must be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from 0 to 1.
2. Select the block(s) to be erased by writing 1's to the appropriate register(s) in LMSR or HSR registers.
If the shadow block is to be erased, this step may be skipped, and LMSR and HSR are ignored. Note that Lock and Select are independent. If a block is selected and locked, no erase will occur.
3. Write to any address in flash. This is referred to as an erase interlock write.
4. Write a logic 1 to the MCR.EHV bit to start the internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR.DONE bit goes high.
6. Confirm MCR.PEG=1.
7. Write a logic 0 to the MCR.EHV bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR.ERS bit to terminate the erase operation.

Additional considerations:

- After setting MCR.ERS, one write, referred to as an interlock write, must be performed before MCR.EHV can be set to 1.
- Data words written during erase sequence interlock writes are ignored.
- An application may terminate the erase sequence by clearing ERS before setting EHV.
- An erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low.
- An erase abort forces the Module to step 8 of the erase sequence.
- An aborted erase will result in MCR.PEG being set low, indicating a failed operation.
- MCR.DONE must be checked to know when the aborting command has completed.
- The block(s) being operated on before the abort contain indeterminate data. This may be recovered by executing an erase on the affected blocks.
- An application may not abort an erase sequence while in erase suspend.

The following example selects two blocks using the LSEL[2-1] bits of the LMSR register to select blocks 2a and 1b (see [Table 137](#) for the flash space memory map and [Section 18.3.6.2, “Low/Mid Address Space Block Locking register \(LML\)”](#),) and performs an erase.

Example 18-2. Erase of Blocks 2a and 1b

```

MCR = 0x00000004; /* Set ERS in MCR: Select ERS Operation */
LMSR = 0x00000006; /* Set LSEL2-1 in LMSR: Select blocks to erase */
(0x000000) = 0xFFFFFFFF; /* Latch a Flash Address with any data */
MCR = 0x00000005; /* Set EHV in MCR: Operation Start */
do /* Loop to wait for DONE=1 */
{ tmp = MCR; /* Read MCR */
} while ( !(tmp & 0x00000400) );
status = MCR & 0x00000200; /* Check PEG flag */
MCR = 0x00000004; /* Reset EHV in MCR: Operation End */
MCR = 0x00000000; /* Reset ERS in MCR: Deselect Operation */

```

18.3.7.1.3 Erase Suspend/Resume

The erase sequence may be suspended to allow read access to the flash array. It is not possible to program or to erase during an erase suspend. During erase suspend, all reads to blocks targeted for erase return indeterminate data.

An erase suspend is initiated by changing the value of the MCR.ESUS bit from 0 to 1. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0 to 1 transition of MCR.ESUS causes the array module to start the sequence which places it in erase suspend.

An application must wait until MCR.DONE=1 before the erase operation is suspended and further actions are attempted. MCR.DONE will go high after MCR.ESUS is set to 1.

Once suspended, the array may be read. Reads while MCR.ESUS=1 from the block(s) being erased return indeterminate data.

Example 18-3. Block Erase Suspend.

```

MCR = 0x00000007; /* Set ESUS in MCR: Erase Suspend */
do /* Loop to wait for DONE=1 */
{ tmp = MCR; /* Read MCR */
} while ( !(tmp & 0x00000400) );

```

Note that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during erase suspend. The erase sequence is resumed by writing a logic 0 to MCR.ESUS.

MCR.EHV must be set to 1 before MCR.ESUS can be cleared to resume the operation.

The array module continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Example 18-4. Block Erase Resume.

```

MCR = 0x00000005; /* Reset ESUS in MCR: Erase Resume */

```

18.3.7.1.4 User Test Mode

User Test Mode is a mode that customers can put the flash array module in to do specific tests to check integrity.

Three kinds of test can be performed:

- Array Integrity Self Check
- Factory Margin Mode Read
- ECC Logic Check

The User Test Mode is equivalent to a modify operation: read accesses attempted during User Test Mode generate a Read-While-Write Error (RWE of MCR set).

User Test operations are not allowed on the Test and Shadow blocks.

Array Integrity Self Check

Array Integrity is checked using a pre-defined address sequence (proprietary), and is executed on selected and unlocked blocks. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UMISR0-4), to determine if an incorrect read, or ECC detection was noted.

The internal MISR calculator is a 32 bit register.

The 128-bit data, the 16 ECC data and the single and double ECC errors of the two double words are therefore captured by the MISR through 5 different read accesses at the same location.

The whole check is done through 5 complete scans of the memory address space:

1. The first pass will scan only bits 31-0 of each page.
2. The second pass will scan only bits 63-32 of each page.
3. The third pass will scan only bits 95-64 of each page.
4. The fourth pass will scan only bits 127-96 of each page.
5. The fifth pass will scan only the ECC bits (8 + 8) and the single and double ECC errors (2 + 2) of both double words of each page.

The 128 data bit and the 16 ECC data are sampled before the eventual ECC correction, while the single and double error flags are sampled after the ECC evaluation.

Only data from existing and unlocked locations are captured by the MISR.

The MISR can be seeded to any value by writing the UMISR0-4 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in LMSR or HSR registers.
Note that Lock and Select are independent. If a block is selected and locked, no Array Integrity Check will occur.
3. Set UT0.AIS bit for a sequential addressing only.
4. Write a logic 1 to the UT0.AIE bit to start the Array Integrity Check.
5. Wait until the UT0.AID bit goes high.
6. Compare UMISR0-4 content with the expected result.
7. Write a logic 0 to the UT0.AIE bit.

8. If more blocks are to be checked, return to step 2.

It is recommended to leave UT0.AIS at 0 and use the proprietary address sequence that checks the read path more fully, although this sequence takes more time. While UT0.AID is low and UT0.AIE is high, an application may clear AIE, resulting in a Array Integrity Check abort. UT0.AID must be checked to know when the aborting command has completed.

The following example selects two blocks using the LSEL[2-1] bits of the LMSR register to select blocks 2a and 1b and performs an array integrity check of those blocks.

Example 18-5. Array Integrity Check of Blocks 2a and 1b

```

UT0 = 0xF9F99999; /* Set UTE in UT0: Enable User Test */
LMSR = 0x00000006; /* Set LSEL2-1 in LMSR: Select blocks */
UT0 = 0x80000002; /* Set AIE in UT0: Operation Start */
do /* Loop to wait for AID=1 */
{ tmp = UT0; /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0 = UMISR0; /* Read UMISR0 content*/
data1 = UMISR1; /* Read UMISR1 content*/
data2 = UMISR2; /* Read UMISR2 content*/
data3 = UMISR3; /* Read UMISR3 content*/
data4 = UMISR4; /* Read UMISR4 content*/
UT0 = 0x00000000; /* Reset UTE and AIE in UT0: Operation End */

```

Factory Margin Read

NOTE

Factory margin read is a diagnostic to check proper programming, for example by 3rd party programming service providers. It is not supported in customer applications because the voltages used for margin reads can reduce the life expectancy of the flash array.

The factory margin read procedure (either Margin 0 or Margin 1) can be run on unlocked blocks to unbalance the sense amplifiers with respect to standard read conditions so that all read accesses reduce the margin vs '0' (UT0.MRV = '0') or vs '1' (UT0.MRV = '1'). Locked sectors are ignored by MISR calculation and ECC flagging.

The results of the factory margin reads can be checked by comparing the checksum value in the UMISR0-4 registers.

Since factory margin reads are done at voltages that are higher than the normal read voltages, lifetime expectancy of the flash may be impacted. Doing factory margin reads repeatedly results in degradation of the flash array and shortens the lifetime expected with normal read levels. For these reasons this capability is reserved for factory use only and is not supported in user applications. Charge losses detected via margin reads are not considered failures of the device and no Failure Analysis will be opened on them.

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in LMS or HBS registers.

Note that Lock and Select are independent. If a block is selected and locked, no Margin Read will occur.

3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Change the value in the UT0.MRE bit from 0 to 1.
5. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
6. Write a logic 1 to the UT0.AIE bit to start the Margin Read Setup or skip to step 6 to terminate.
7. Wait until the UT0.AID bit goes high.
8. Compare UMISR0-4 content with the expected result.
9. Write a logic 0 to the UT0.AIE, UT0.MRE and UT0.MRV bits.
10. If more blocks are to be checked, return to step 2.

It is recommended to leave UT0.AIS at 1 and use the linear address sequence, which takes less time.

During the execution of the Margin Read operation it is forbidden to modify the content of Block Select (LMS, HBS) and Lock (LML, SLL, HBL) registers, otherwise the MISR value can vary in an unpredictable way.

The read accesses will be done with the addition of a proper number of Wait States to guarantee the correctness of the result.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in a Array Integrity Check abort.

UT0.AID must be checked to know when the aborting command has completed.

Example 18-6. Margin Read Check versus 1's .

```

UMISR0 = 0x00000000; /* Reset UMISR0 content */
UMISR1 = 0x00000000; /* Reset UMISR1 content */
UMISR2 = 0x00000000; /* Reset UMISR2 content */
UMISR3 = 0x00000000; /* Reset UMISR3 content */
UMISR4 = 0x00000000; /* Reset UMISR4 content */
UT0 = 0xF9F99999; /* Set UTE in UT0: Enable User Test */
LMS = 0x00000006; /* Set LSL2-1 in LMS: Select Sectors */
UT0 = 0x80000004; /* Set AIS in UT0: Select Operation */
UT0 = 0x80000024; /* Set MRE in UT0: Select Operation */
UT0 = 0x80000034; /* Set MRV in UT0: Select Margin versus 1's */
UT0 = 0x80000036; /* Set AIE in UT0: Operation Start */
do /* Loop to wait for AID=1 */
{ tmp = UT0; /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0 = UMISR0; /* Read UMISR0 content*/
data1 = UMISR1; /* Read UMISR1 content*/
data2 = UMISR2; /* Read UMISR2 content*/
data3 = UMISR3; /* Read UMISR3 content*/
data4 = UMISR4; /* Read UMISR4 content*/
UT0 = 0x80000034; /* Reset AIE in UT0: Operation End */
UT0 = 0x00000000; /* Reset UTE, MRE, MRV, AIS in UT0: Deselect Op. */
    
```

ECC Logic Check

ECC Logic Check verifies the integrity of the ECC correction and detection logic. The operation provides user control over the 64 data bit + 8 parity bit inputs. Results of the ECC logic can be checked by reading the MISR value.

The ECC Logic Check operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Write in UT1.DAI31-0 and UT2.DAI63-32 the double word input value.
3. Write in UT0.DSI7-0 the Syndrome Input value.
4. Select the ECC Logic Check: write a logic 1 to the UT0.EIE bit.
5. Write a logic 1 to the UT0.AIE bit to start the ECC Logic Check.
6. Wait until the UT0.AID bit goes high.
7. Compare UMISR0-4 content with the expected result.
8. Write a logic 0 to the UT0.AIE bit.

Notice that when UT0.AID is low UMISR0-4, UT1-2 and bits MRE, MRV, EIE, AIS and DSI7-0 of UT0 are not accessible: reading returns indeterminate data and writing has no effect.

Example 18-7. ECC Logic Check

```

UT0 = 0xF9F99999; /* Set UTE in UT0: Enable User Test */
UT1 = 0x55555555; /* Set DAI31-0 in UT1: Even Word Input Data */
UT2 = 0xAAAAAAAA; /* Set DAI63-32 in UT2: Odd Word Input Data */
UT0 = 0x80FF0000; /* Set DSI7-0 in UT0: Syndrome Input Data */
UT0 = 0x80FF0008; /* Set EIE in UT0: Select ECC Logic Check */
UT0 = 0x80FF000A; /* Set AIE in UT0: Operation Start */
do /* Loop to wait for AID=1 */
{ tmp = UT0; /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0 = UMISR0; /* Read UMISR0 content (expected 0x55555555) */
data1 = UMISR1; /* Read UMISR1 content (expected 0xAAAAAAAA) */
data2 = UMISR2; /* Read UMISR2 content (expected 0x55555555) */
data3 = UMISR3; /* Read UMISR3 content (expected 0xAAAAAAAA) */
data4 = UMISR4; /* Read UMISR4 content (expected 0x00FF00FF) */
UT0 = 0x00000000; /* Reset UTE, AIE and EIE in UT0: Operation End */

```

18.3.7.2 Error correction code

The Flash module provides a method to improve the reliability of the data stored in Flash: the usage of an Error Correction Code. The word size is fixed at 64 bits.

Eight ECC bits, programmed to guarantee a Single Error Correction and a Double Error Detection (SEC-DED), are associated to each 64-bit Double Word.

ECC circuitry provides correction of single bit faults and is used to achieve automotive reliability targets. Some units will experience single bit corrections throughout the life of the product with no impact to product reliability.

18.3.7.2.1 ECC algorithms

The Flash module supports one ECC Algorithm: “All ‘1’s No Error”. A modified Hamming code is used that ensures the all erased state (that is, 0xFFFF....FFFF) data is a valid state, and will not cause an ECC error. This allows the user to perform a blank check after a sector erase operation.

18.3.7.3 EEprom emulation

18.3.7.4 Eprom Emulation

The chosen ECC algorithm allows some bit manipulations so that a Double Word can be rewritten several times without needing an erase of the sector. This allows to use a Double Word to store flags useful for the Eeprom Emulation. As an example the chosen ECC algorithm allows to start from an All ‘1’s Double Word value and rewrite whichever of its four 16-bits Half-Words to an All ‘0’s content by keeping the same ECC value.

The following table shows a set of Double Words sharing the same ECC value:

Table 176. Bits Manipulation: Double Words with the same ECC value

Double Word	ECC All ‘1’s No Error
0xFFFF_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_FFFF_0000	0xFF
0xFFFF_FFFF_0000_FFFF	0xFF
0xFFFF_0000_FFFF_FFFF	0xFF
0x0000_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_0000_0000	0xFF
0xFFFF_0000_FFFF_0000	0xFF
0x0000_FFFF_FFFF_0000	0xFF
0xFFFF_0000_0000_FFFF	0xFF
0x0000_FFFF_0000_FFFF	0xFF
0x0000_0000_FFFF_FFFF	0xFF
0xFFFF_0000_0000_0000	0xFF
0x0000_FFFF_0000_0000	0xFF
0x0000_0000_0000_0000	0xFF

When some Flash sectors are used to perform an Eeprom Emulation, it is recommended for safety reasons to reserve at least 3 sectors to this purpose.

18.3.7.4.1 All ‘1’s No Error

The All ‘1’s No Error Algorithm detects as valid any Double Word read on a just erased sector (all the 72 bits are ‘1’s).

This option allows to perform a Blank Check after a Sector Erase operation.

18.3.7.5 Protection strategy

Two kinds of protection are available: Modify Protection to avoid unwanted program/erase in Flash sectors and Censored Mode to avoid piracy.

18.3.7.5.1 Modify protection

The Flash Modify Protection information is stored in non-volatile Flash cells located in the TestFlash. This information is read once during the Flash initialization phase following the exiting from Reset and is stored in volatile registers that act as actuators.

The reset state of all the volatile modify protection registers is the protected state.

All the non-volatile modify protection registers can be programmed through a normal Double Word Program operation at the related locations in TestFlash.

The non-volatile modify protection registers cannot be erased.

- The non-volatile Modify Protection Registers are physically located in TestFlash their bits can be programmed to '0' only once and they can no more be restored to '1'.
- The Volatile Modify Protection Registers are Read/Write registers which bits can be written at '0' or '1' by the user application.

A software mechanism is provided to independently lock/unlock each Low, Mid and High Address Space Block against program and erase.

Software locking is done through the LML (Low/Mid Address Space Block Lock Register) or HBL (High Address Space Block Lock Register) registers.

An alternate means to enable software locking for blocks of Low Address Space only is through the SLL (Secondary Low/Mid Address Space Block Lock Register).

All these registers have a non-volatile image stored in TestFlash (NVLML, NVHBL, NVSLL), so that the locking information is kept on reset.

On delivery the TestFlash non-volatile image is at all '1's, meaning all sectors are locked.

By programming the non-volatile locations in TestFlash the selected sectors can be unlocked.

Being the TestFlash One Time Programmable (that is, not erasable), once unlocked the sectors cannot be locked again.

Of course, on the contrary, all the volatile registers can be written at 0 or 1 at any time, therefore the user application can lock and unlock sectors when desired.

18.3.7.5.2 Censored Mode

The Censored Mode information is stored in non-volatile Flash cells located in the Shadow Sector. This information is read once during the Flash initialization phase following the exiting from Reset and is stored in volatile registers that act as actuators.

The reset state of all the Volatile Censored Mode Registers is the protected state.

All the non-volatile Censored Mode registers can be programmed through a normal Double Word Program operation at the related locations in the Shadow Sector.

The non-volatile Censored Mode registers can be erased by erasing the Shadow Sector.

- The non-volatile Censored Mode Registers are physically located in the Shadow Sector their bits can be programmed to '0' and eventually restored to '1' by erasing the Shadow Sector.
- The Volatile Censored Mode Registers are registers not accessible by the user application.

The Flash module provides two levels of protection against piracy:

- If bits CW15:0 of NVSCI0 are programmed at 0x55AA and NVSC1 = NVSCI0 the Censored Mode is disabled, while all the other possible values enable the Censored Mode.
- If bits SC15:0 of NVSCI0 are programmed at 0x55AA and NVSC1 = NVSCI0 the Public Access is disabled, while all the other possible values enable the Public Access.

The parts are delivered to the user with Censored Mode and Public Access disabled.

18.4 Data Flash Memory

18.4.1 Block Overview

The primary function of the Flash Module is to serve as electrically programmable and erasable Non-Volatile Memory.

NV Memory may be used for instruction and/or data storage.

The Module is a Non-Volatile solid-state silicon memory device consisting of blocks (called also sectors) of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The Flash Module is arranged as two functional units: the Flash Core and the Memory Interface.

The Flash Core is composed of arrayed Non-Volatile storage elements, sense amplifiers, row decoders, column decoders and charge pumps. The arrayed storage elements in the Flash Core are sub-divided into physically separate units referred to as blocks (or sectors).

Flash core is organized including ECC correction code. ECC circuitry provides correction of single bit faults and is used to achieve automotive reliability targets. Some units will experience single bit corrections throughout the life of the product with no impact to product reliability.

The Memory Interface contains the registers and logic which control the operation of the Flash Core. The Memory Interface is also the interface between the Flash Module and a Bus Interface Unit (BIU) and may contain the ECC logic and redundancy logic.

A BIU connects the Flash Module to a system bus, and contains all system level customization required for the SoC application. The Flash Module is generic and requires a BIU to configure it for different SoC applications. A BIU is not included as a part of the Flash Module.

18.4.2 Features

- 120 ns Access Time
- 32 bits Read/Write parallelism
- 7 bits Error Correction Code (SEC-DED) to enhance Data Retention
- Sector Erase
- Single Bank: Read-While-Modify not available
- Erase Suspend available (Program Suspend not available)
- Software programmable Program/Erase Protection to avoid unwanted writings
- Shadow Sector not available
- Optimized Data Flash is a slave IP that requires clocks and reference current coming from Master LC Data Flash

18.4.3 Block Diagram

The Flash Macrocell contains one Matrix Module, composed by a Single Bank: Bank 0, normally used for Code storage. No Read-While-Modify operations are possible.

The Modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are given through a User Registers Interface. The read data bus is 32 bits wide, while the Flash registers are on a separate bus 32 bits wide. The High Voltages needed for Program/Erase operations are internally generated.

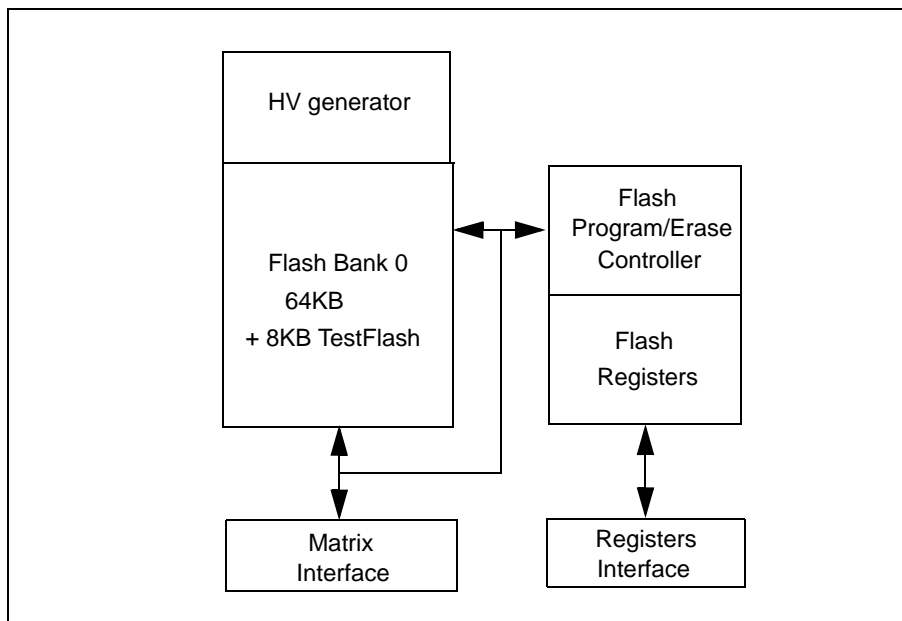


Figure 171. Flash Macrocell Structure

18.4.4 Functional Description

18.4.4.1 Macrocell Structure

The Flash Macrocell is designed for use in embedded MCU/SoC applications which require Data Non-Volatile Memories for EE emulation.

The Flash Module is addressable by Word (32 bits) for program and for read.

The Flash Module supports fault tolerance through Error Correction Code (ECC) and/or error detection. The ECC implemented within the Flash Module will correct single bit failures and detect double bit failures.

The Flash Module uses an embedded hardware algorithm implemented in the Memory Interface to program and erase the Flash Core.

Control logic that works with the software block enables, and software lock mechanisms, is included in the embedded hardware algorithm to guard against accidental program/erase.

The hardware algorithm perform the steps necessary to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the Flash Module reads as logic level 0 (or low).

An erased bit in the Flash Module reads as logic level 1 (or high).

Program and erase of the Flash Module requires multiple system clock cycles to complete.

The erase sequence may be suspended.

The program and erase sequences may be aborted.

Being a slave IP, Data Flash requires Code Flash to be active (means not under reset or in Disable Mode or in Sleep Mode) in order to be active.

18.4.4.2 Data flash sectorization

The Flash Module supports memory sizes of 72 KB of User Memory, plus 8KB of Test Memory.

There are two User Address Spaces: Low and Mid Address Space.

There is only one size of blocks available to the User in the Flash Core: 16KB. 8KB is reserved for Test Flash

The Flash Module is composed by a single Bank (Bank 0): Read-While-Modify is not supported.

Bank 0 of the 72 KB Flash macrocell is divided in 4 sectors. Bank 0 contains also a reserved sector named TestFlash in which some One Time Programmable User data are stored.

Table 177. Data Flash Module Sectorization

Bank	Sector	Addresses	Size	Address Space
B0	B0F0	0x000000 to 0x003FFF	16KB	Low Address Space
B0	B0F1	0x004000 to 0x007FFF	16KB	Low Address Space
B0	B0F2	0x008000 to 0x00BFFF	16KB	Low Address Space
B0	B0F3	0x00C000 to 0x00FFFF	16KB	Low Address Space

Table 177. Data Flash Module Sectorization

Bank	Sector	Addresses	Size	Address Space
B0	Reserved	0x010000 to 0x03FFFF	192KB	Low Address Space
B0	Reserved	0x040000 to 0x07FFFF	256KB	Mid Address Space
B0	B0TF	0x402000 to 0x403FFF	8KB	Test Address Space
B0	Reserved	0x404000 to 0x7FFFFFFF	4080KB	Test Address Space

The Flash Module is divided into blocks also to implement independent Erase/Program protection. A software mechanism is provided to independently lock/unlock each block in low, mid address space against program and erase.

18.4.4.3 Test Flash Block

The TestFlash block exists outside the normal address space and is programmed, erased and read independently of the other blocks. The independent TestFlash block is reserved to store the Non Volatile informations related to Redundancy, Configuration and Protection.

Due to this special usage, the TestFlash sector is not affected by the Column Redundancy. The ECC, on the contrary, is applied also to TestFlash.

The usage of reserved TestFlash sector is detailed in the following table.

Table 178. TestFlash structure

Name	Description	Addresses	Size
	User Reserved	0x403D00 to 0x403DE7	232 byte
NVLML	NV Low/Mid address space block Locking reg	0x403DE8 to 0x403DEF	8 byte
	Reserved	0x403DF0 to 0x403DF7	8 byte
NVSLL	NV Secondary Low/mid add space block Lock reg	0x403DF8 to 0x403DFF	8 byte
	User Reserved	0x403E00 to 0x403EFF	256 byte
	Reserved	0x403F00 to 0x403FB7	184 byte

The Test Flash block can be enabled by the BIU. When the Test space is enabled, the program operations to the Test block are allowed from 0x403D00 to 0x403EFF (User/Lock area is One Time Programmable). User Mode program of the test block are enabled only when MCR.PEAS is high. The TestFlash block contains specified data that are needed for Flash Macrocell or the device features.

In User Mode the Flash Module may be read and written (register writes and interlock writes), programmed or erased. The default state of the Flash Module is read. The main and test address space can be read only in the read state.

The Flash registers are always available for read, also when the Module is in disable mode (except few documented registers). The Flash Module enters the read state on reset. The Module is in the read state under two sets of conditions:

- The read state is active when the Module is enabled (User Mode Read)

- The read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend).

Notice that no Read-While-Modify is available. Flash Core reads return 32 bits. Registers reads return 32 bits (1 Word). Flash Core reads are done through the Bus Interface Unit.

Registers reads to unmapped register address space will return all 0's. Registers writes to unmapped register address space will have no effect. Array reads attempted to invalid locations will result in indeterminate data. Invalid locations occur when addressing is done to blocks that do not exist in non 2n array sizes.

Interlock writes attempted to invalid locations, will result in an interlock occurring, but attempts to program these blocks will not occur since they are forced to be locked. Erase will occur to selected and unlocked blocks even if the interlock write is to an invalid location.

Simultaneous Read cycle on the Flash Matrix and Read/Write cycles on the Registers are possible. On the contrary Registers Read/Write accesses simultaneous to a Flash Matrix interlock write are forbidden.

18.4.4.4 Reset

A reset is the highest priority operation for the Flash module and terminates all other operations.

The Flash Module uses reset to initialize register and status bits to their default reset values. If the Flash Module is executing a Program or Erase operation (MCR.PGM = 1 or MCR.ERS = 1) and a reset is issued, the operation will be suddenly terminated and the module will disable the high voltage logic without damage to the high voltage circuits. Reset terminates all operations and forces the Flash Module into User mode ready to receive accesses. Reset and power-off must not be used as a systematic way to terminate a Program or Erase operation.

After reset is negated, read register access may be done, although it should be noted that registers that require updating from TEST block or KRAM information, or other inputs, may not be read until MCR.DONE transitions. MCR.DONE may be polled to determine if the Flash module has transitioned out of reset. Notice that the registers cannot be written until MCR.DONE is high.

18.4.4.5 Power-down mode

The power-down mode allows to turn off all Flash DC current sources, so that all power dissipation is due only to leakage in this mode.

Reads from or writes to the module are not possible in power-down mode.

The user may not read some registers (UMISR0–1, UT1–1 and part of UT0) until the power-down mode is exited. On the contrary write access is locked on all the registers in Disable Mode.

When enabled the Flash Module returns to its pre-disable state in all cases unless in the process of executing an erase high voltage operation at the time of disable.

If the Flash Module is disabled during an erase operation, MCR.ESUS bit is set to 1. This means that Flash macrocell is first put into suspend state (after t_{SUSP}). The User may resume the erase operation at the time the Module is enabled by clearing MCR.ESUS bit. MCR.EHV must be high to resume the erase operation.

If the Flash Module is disabled during a program operation, the Disable Mode will be entered only after the programming ends.

18.4.4.6 Slave Mode

Being a slave, Data Flash requires Code Flash to be active (means not under reset or in Disable Mode or in Sleep Mode) in order to be active.

It is forbidden to put code flash0 in Disable Mode or in Sleep mode or under reset when the data flash is active.

18.4.5 Register description

The Flash user registers mapping is shown in the [Table 179](#).

Table 179. Data Flash Registers

Address offset	Register name
0x0000	Module Configuration Register (MCR)
0x0004	Low/Mid Address Space Block Locking register (LML)
0x0008	Reserved
0x000C	Secondary Low/Mid Address Space Block Locking register (SLL)
0x0010	Low/Mid Address Space Block Select register (LMS)
0x0014	Reserved
0x0018	Address Register (ADR)
0x001C-0x0038	Reserved
0x003C	User Test 0 register (UT0)
0x0040	User Test 1 register (UT1)
0x0044	Reserved
0x0048	User Multiple Input Signature Register 0 (UMISR0)
0x004C	User Multiple Input Signature Register 1 (UMISR1)
0x0050-0x0058	Reserved

Locations 0x0044, 0x0050, 0x0054 and 0x0058 are Write/Read from user point of view but no functionality is associated. Registers are not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.

In the following some non-volatile registers are described. Please notice that such entities are not Flip-Flops, but locations of TestFlash sector with a special meaning.

During the Flash initialization phase, the FPEC reads these non-volatile registers and update the corresponding volatile registers. When the FPEC detects ECC double errors in these special locations, it behaves in the following way:

- In case of a failing system locations (configurations, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In case of failing user locations (protections, ...), the volatile registers are filled with all ‘1’s and the Flash initialization ends setting low the PEG bit of MCR.

Table 180 lists bit access type abbreviations used in this section.

Table 180. Abbreviations

Abbreviation	Case	Description
rw	read/write	The software can read and write to these bits.
rc	read/clear	The software can read and clear to these bits.
r	read-only	The software can only read these bits.
w	write-only	The software should only write to these bits.

18.4.5.1 Module Configuration Register (MCR)

The Module Configuration Register enables and monitors all the modify operations of each flash module. Identical MCRs are provided in the data flash blocks.

Address: Base + 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EDC	0	0	0	0	SIZE2	SIZE1	SIZE0	0	LAS2	LAS1	LAS0	0	MAS2	MAS1	MAS0
W	r1c															
Reset	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	0	0	PEAS	DONE	PEG	0	0	0	0	0	PGM	PSUS	ERS	ESUS
W	r1c	r1c														
Reset	0	0	0	0	0	X	1	0	0	0	0	0	0	0	0	0

Figure 172. Module Configuration Register (MCR)

Table 181. MCR field descriptions

Field	Description
EDC	<p>EDC: Ecc Data Correction (Read/Clear)</p> <p>EDC provides information on previous reads. If a ECC Single Error detection and correction occurred, the EDC bit will be set to 1. This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the User.</p> <p>In the event of a ECC Double Error detection, this bit will not be set.</p> <p>If EDC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EDC) were not corrected through ECC.</p> <p>Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect.</p> <p>The function of this bit is SoC dependent and it can be configured to be disabled.</p> <p>0: Reads are occurring normally.</p> <p>1: An ECC Single Error occurred and was corrected during a previous read.</p>

Table 181. MCR field descriptions (continued)

Field	Description
SIZE[2:0]	<p>Array space SIZE 2–0</p> <p>The value of SIZE field depends on the size of the flash module: 110 64 KB</p>
LAS[2:0]	<p>Low Address Space 2–0</p> <p>The value of the LAS field corresponds to the configuration of the Low Address Space: 110 4 × 16 KB</p>
MAS[2:0]	<p>Mid Address Space</p> <p>The value of the MAS field corresponds to the configuration of the Mid Address Space:</p>
EER	<p>EER: ECC event Error (Read/Clear)</p> <p>EER provides information on previous reads. If an ECC Double Error detection occurred, the EER bit is set to '1'.</p> <p>This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to '1' by the user.</p> <p>In the event of an ECC Single Error detection and correction, this bit will not be set.</p> <p>If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) were correct.</p> <p>Since this bit is an error flag, it must be cleared to '0' by writing 1 to the register location. A write of 0 will have no effect.</p> <p>0: Reads are occurring normally. 1: An ECC Double Error occurred during a previous read.</p>
RWE	<p>RWE: Read-while-Write event Error (Read/Clear)</p> <p>RWE provides information on previous reads when a Modify operation is on going. If a RWW Error occurs, the RWE bit will be set to 1. Read-While-Write Error means that a read access to the Flash Matrix has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check.</p> <p>This bit must then be cleared, or a reset must occur before this bit will return to a 0 state. This bit may not be set to 1 by the User.</p> <p>If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) were correct.</p> <p>Since this bit is an error flag, it must be cleared to 0 by writing 1 to the register location. A write of 0 will have no effect.</p> <p>0: Reads are occurring normally. 1: A RWW Error occurred during a previous read.</p>
PEAS	<p>PEAS: Program/Erase Access Space (Read Only)</p> <p>PEAS is used to indicate which space is valid for program and erase operations: main array space or test space.</p> <p>PEAS = 0 indicates that the main address space is active for all Flash module program and erase operations.</p> <p>PEAS = 1 indicates that the test address space is active for program and erase.</p> <p>The value in PEAS is captured and held with the first interlock write done for Modify operations. The value of PEAS is retained between sampling events (that is, subsequent first interlock writes).</p> <p>0: Test address space is disabled for program/erase and main address space enabled. 1: Test address space is enabled for program/erase and main address space disabled.</p>

Table 181. MCR field descriptions (continued)

Field	Description
DONE	<p>DONE: <i>modify operation DONE</i> (Read Only)</p> <p>DONE indicates if the Flash Module is performing a high voltage operation. DONE is set to 1 on termination of the Flash Module reset. DONE is cleared to 0 just after a 0 to 1 transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. DONE is set to 1 at the end of program and erase high voltage sequences. DONE is set to 1 (within tPABT or tEABT, equal to P/E Abort Latency) after a 1 to 0 transition of EHV, which aborts a high voltage Program/Erase operation. DONE is set to 1 (within tESUS, time equals to Erase Suspend Latency) after a 0 to 1 transition of ESUS, which suspends an erase operation.</p> <p>0: Flash is executing a high voltage operation. 1: Flash is not executing a high voltage operation.</p>
PEG	<p>PEG: <i>Program/Erase Good</i> (Read Only)</p> <p>The PEG bit indicates the completion status of the last Flash Program, Erase, AIC or MM sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the Program, Erase, AIC or MM high voltage operations. Aborting a Program/Erase/AIC/MM high voltage operation will cause PEG to be cleared to '0', indicating the sequence failed. PEG is set to '1' when the Flash Module is reset, unless a Flash initialization error has been detected. The value of PEG is valid only when PGM=1 and/or ERS=1 and after DONE transitions from '0' to '1' due to an abort or the completion of a Program/Erase/AIC/MM operation. PEG is valid until PGM/ERS makes a '1' to '0' transition or EHV makes a '0' to '1' transition. The value in PEG is not valid after a '0' to '1' transition of DONE caused by ESUS being set to logic '1'. If Program or Erase are attempted on blocks that are locked, the response will be PEG=1, indicating that the operation was successful, and the content of the block were properly protected from the Program or Erase operation. If a Program operation tries to program at '1' bits that are at '0', the program operation is correctly executed on the new bits to be programmed at '0', but PEG is cleared, indicating that the requested operation has failed. In AIC or MM PEG is set to '1' when the operation is completed, regardless the occurrence of any error. The presence of errors can be detected only comparing checksum value stored in UMIRS0-1.</p> <p>0: Program or Erase, operation failed or aborted. 1: Program or Erase operation successful. 0: AIC or MM aborted. 1: AIC or MM operation successfully concluded, with or without checksum errors.</p>
PGM	<p>PGM: <i>ProGraM</i> (Read/Write)</p> <p>PGM is used to set up the Flash module for a Program operation. A 0 to 1 transition of PGM initiates a Program sequence. A 1 to 0 transition of PGM ends the Program sequence. PGM can be set only under User Mode Read (ERS is low and UT0.AIE is low). PGM can be cleared by the user only when EHV is low and DONE is high. PGM is cleared on reset.</p> <p>0: Flash is not executing a Program sequence. 1: Flash is executing a Program sequence.</p>
PSUS	<p>PSUS: <i>Program SUSpend</i> (Read/Write)</p> <p>Write this bit has no effect, but the written data can be read back.</p>

Table 181. MCR field descriptions (continued)

Field	Description
ERS	<p>ERS: ERaSe (Read/Write) ERS is used to set up the Flash module for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can be set only under User Mode Read (PGM is low and UT0.AIE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0: Flash is not executing an erase sequence. 1: Flash is executing an erase sequence.</p>
ESUS	<p>ESUS: Erase SUSpend (Read/Write) ESUS is used to indicate that the Flash module is in Erase Suspend or in the process of entering a Suspend state. The Flash module is in Erase Suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the Flash in Erase Suspend. The Flash module enters Suspend within t_{ESUS} of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the module to Erase. The Flash module cannot exit Erase Suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0: Erase sequence is not suspended. 1: Erase sequence is suspended.</p>
EHV	<p>EHV: Enable High Voltage (Read/Write) The EHV bit enables the Flash Module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a Program/Erase sequence. EHV may be set under one of the following conditions: Erase (ERS=1, ESUS=0, UT0.AIE=0) Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0)</p> <p>In normal operation, a 1 to 0 transition of EHV with DONE high and ESUS low terminates the current Program/Erase high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the eventual Suspend bit low. An abort causes the value of PEG to be cleared, indicating a failing Program/Erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. A suspended operation cannot be aborted. Aborting a high voltage operation will leave the Flash Module addresses in an undeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0: Flash is not enabled to perform an high voltage operation. 1: Flash is enabled to perform an high voltage operation.</p>

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit by bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The flash module does not allow the user to write bits simultaneously which would put the device into an illegal state. This is implemented through a priority mechanism among the bits. [Table 151](#) shows the bit changing priorities.

Table 182. MCR bits set/clear priority levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS

If the user attempts to write two or more MCR bits simultaneously, only the bit with the lowest priority level is written.

18.4.5.2 Low/Mid Address Space Block Locking register (LML)

The Low/Mid Address Space Block Locking register provides a means to protect blocks from being modified. These bits, along with bits in the SLL register, determine if the block is locked from program or erase. An “OR” of LML and SLL determine the final lock status. Identical LML registers are provided in the code flash and the data flash blocks.

The LML register has a related Non Volatile Low/Mid Address Space Block Locking register located in TestFlash that contains the default reset value for LML: the NVLML register is read during the reset phase of the Flash Module and loaded into the LML.

Address: Base + 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LME	0	0	0	0	0	0	0	0	0	0	TSLK	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LLK	LLK	LLK	LLK
W													3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x

Figure 173. Low/Mid Address Space Block Locking register (LML)

18.4.5.3 Non-Volatile Low/Mid Address Space Block Locking register (NVLML)

Address: Base + 0x40_3DE8

Delivery value:
0xFFFFFFFF

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LME	0	0	0	0	0	0	0	0	0	0	TSLK	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LLK	LLK	LLK	LLK
W													3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x

Figure 174. Non-Volatile Low/Mid Address Space Block Locking register (NVLML)

The NVLML register is a 64-bit register, the 32 most significant bits of which (bits 63:32) are “don’t care” bits that are eventually used to manage ECC codes. Identical NVLML registers are provided in the code flash and the data flash blocks.

Table 183. LML /NVLML field descriptions

Field	Description
LME ¹	<p>LME: Low/Mid address space block Enable (Read Only) This bit is used to enable the Lock registers (TSLK and LLK3-0) to be set or cleared by registers writes.</p> <p>This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the LME bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For LME the password 0xA1A11111 must be written to the LML register.</p> <p>0: Low Address Locks are disabled: TSLK and LLK3-0 cannot be written. 1: Low Address Locks are enabled: TSLK and LLK3-0 can be written.</p>
TSLK	<p>TSLK: Test address space block Lock (Read/Write) This bit is used to lock the block of Test Address Space from Program and Erase (Erase is any case forbidden for Test block).</p> <p>A value of 1 in the TSLK register signifies that the Test block is locked for Program and Erase. A value of 0 in the TSLK register signifies that the Test block is available to receive Program and Erase pulses.</p> <p>The TSLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the TSLK register is not writable if a high voltage operation is suspended or if a margin mode is on going.</p> <p>Upon reset, information from the TestFlash block is loaded into the TSLK register. The TSLK bit may be written as a register. Reset will cause the bit to go back to its TestFlash block value. The default value of the TSLK bit (assuming erased fuses) would be locked.</p> <p>TSLK is not writable unless LME is high.</p> <p>0: Test Address Space Block is unlocked and can be modified (if also SLL.STSLK=0). 1: Test Address Space Block is locked and cannot be modified.</p>

Table 183. LML /NVLML field descriptions (continued)

Field	Description
LLK[3:0]	<p>LLK3-0: Low address space block Lock 3-0 (Read/Write) These bits are used to lock the blocks of Low Address Space from Program and Erase. LLK3-0 are related to sectors B0F3-0, respectively. A value of 1 in a bit of the LLK register signifies that the corresponding block is locked for Program and Erase. A value of 0 in a bit of the LLK register signifies that the corresponding block is available to receive Program and Erase pulses. The LLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the LLK register is not writable if a high voltage operation is suspended or if a margin mode is on going. Upon reset, information from the TestFlash block is loaded into the LLK registers. The LLK bits may be written as a register. Reset will cause the bits to go back to their TestFlash block value. The default value of the LLK bits (assuming erased fuses) would be locked. In the event that blocks are not present (due to configuration or total memory size), the LLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the TestFlash block), and register writes will have no effect. In the 72 KB Flash Macrocell the writability of bits LLK3-0 is controlled by bits CS3-0 of FVSCR. LLK is not writable unless LME is high. 0: Low Address Space Block is unlocked and can be modified (if also SLL.SLK=0). 1: Low Address Space Block is locked and cannot be modified.</p>

¹ This field is present only in LML

18.4.5.4 Secondary Low/Mid Address Space Block Locking register (SLL)

The Secondary Low/Mid Address Space Block Locking register provides an alternative means to protect blocks from being modified. These bits, along with bits in the LML register, determine if the block is locked from program or Erase. An “OR” of LML and SLL determine the final lock status. Identical SLL registers are provided in the code flash and the data flash blocks.

The SLL register has a related Non Volatile Secondary Low/Mid Address Space Block Locking register located in TestFlash that contains the default reset value for SLL: the NVSLL register is read during the reset phase of the Flash Module and loaded into the SLL.

Address: Base + 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	SLE	0	0	0	0	0	0	0	0	0	0	0	STS	0	0	0	0
W													LK				
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	SLK	SLK	SLK	SLK	
W													3	2	1	0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	

Figure 175. Secondary Low/mid address space block Locking reg (SLL)

18.4.5.5 Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL)

The NVSLL register is a 64-bit register, the 32 most significant bits of which (bits 63:32) are “don’t care” bits that are eventually used to manage ECC codes. Identical NVSLL registers are provided in the code flash and the data flash blocks.

Address: Base + 0x40_3DF8

Delivery value:
0xFFFFFFFF

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLE	0	0	0	0	0	0	0	0	0	0	STSLK	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	SLK3	SLK2	SLK1	SLK0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x

Figure 176. Non-Volatile Secondary Low/Mid Address Space Block Locking register (NVSLL)

Table 184. SLL and NVSLL field descriptions

Field	Description
SLE ¹	<p>Secondary Low/Mid Address Space Block Enable</p> <p>This bit is used to enable the Lock registers (STSLK and SLK3-0) to be set or cleared by registers writes.</p> <p>This bit is a status bit only. The method to set this bit is to write a password, and if the password matches, the SLE bit will be set to reflect the status of enabled, and is enabled until a reset operation occurs. For SLE the password 0xC3C33333 must be written to the SLL register.</p> <p>0: Secondary Low/Mid Address Locks are disabled: STSLK and SLK3-0 cannot be written.</p> <p>1: Secondary Low/Mid Address Locks are enabled: STSLK and SLK3-0 can be written.</p>
STSLK	<p>Secondary Test/Shadow address space block Lock</p> <p>This bit is used as an alternate means to lock the block of Test Address Space from Program and Erase (Erase is any case forbidden for Test block).</p> <p>A value of 1 in the STSLK register signifies that the Test block is locked for Program and Erase. A value of 0 in the STSLK register signifies that the Test block is available to receive Program and Erase pulses.</p> <p>The STSLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the STSLK register is not writable if a high voltage operation is suspended or if a margin mode is on going.</p> <p>Upon reset, information from the TestFlash block is loaded into the STSLK register. The STSLK bit may be written as a register. Reset will cause the bit to go back to its TestFlash block value. The default value of the STSLK bit (assuming erased fuses) would be locked.</p> <p>STSLK is not writable unless SLE is high.</p> <p>0: Test Address Space Block is unlocked and can be modified (if also LML.TSLK=0).</p> <p>1: Test Address Space Block is locked and cannot be modified.</p>

Table 184. SLL and NVSLL field descriptions (continued)

Field	Description
SLK[3:0]	<p>Secondary Low Address Space Block Lock 3–0</p> <p>These bits are used as an alternate means to lock the blocks of Low Address Space from Program and Erase.</p> <p>SLK3-0 are related to sectors B0F3-0, respectively.</p> <p>A value of 1 in a bit of the SLK register signifies that the corresponding block is locked for Program and Erase.</p> <p>A value of 0 in a bit of the SLK register signifies that the corresponding block is available to receive Program and Erase pulses.</p> <p>The SLK register is not writable once an interlock write is completed until MCR.DONE is set at the completion of the requested operation. Likewise, the SLK register is not writable if a high voltage operation is suspended or if a margin mode is on going.</p> <p>Upon reset, information from the TestFlash block is loaded into the SLK registers. The SLK bits may be written as a register. Reset will cause the bits to go back to their TestFlash block value. The default value of the SLK bits (assuming erased fuses) would be locked.</p> <p>In the event that blocks are not present (due to configuration or total memory size), the SLK bits will default to locked, and will not be writable. The reset value will always be 1 (independent of the TestFlash block), and register writes will have no effect.</p> <p>SLK is not writable unless SLE is high.</p> <p>0: Low Address Space Block is unlocked and can be modified (if also LML.LLK=0).</p> <p>1: Low Address Space Block is locked and cannot be modified.</p>

¹ This field is present only in SLL

18.4.5.6 Low/Mid Address Space Block Select register (LMS)

The Low/Mid Address Space Block Select register provides a means to select blocks to be operated on during erase. Identical LMS registers are provided in the code flash and the data flash blocks.

Address: Base + 0x0010												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	LSL	LSL	LSL	LSL
W													3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 177. Low/Mid Address Space Block Select register (LMS)

Table 185. LMS field descriptions

Field	Description
LSL[3:0]	<p>Low Address Space Block Select 3–0</p> <p>LSL3-0: Low address space block SeLect 3-0 (Read/Write)</p> <p>A value of 1 in the select register signifies that the block is selected for erase.</p> <p>A value of 0 in the select register signifies that the block is not selected for erase. The reset value for the select register is 0, or unselected.</p> <p>LSL3-0 are related to sectors B0F3-0, respectively.</p> <p>The blocks must be selected (or unselected) before doing an erase interlock write as part of the Erase sequence. The select register is not writable once an interlock write is completed or if a high voltage operation is suspended or if a margin mode is on going.</p> <p>In the event that blocks are not present (due to configuration or total memory size), the corresponding LSL bits will default to unselected, and will not be writable. The reset value will always be 0, and register writes will have no effect.</p> <p>0: Low Address Space Block is unselected for Erase. 1: Low Address Space Block is selected for Erase.</p>

18.4.5.7 Address Register (ADR)

The Address Register provides the first failing address in the event module failures (ECC, RWW or FPEC) or the first address at which a ECC single error correction occurs.

Address: Base + 0x0018

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	AD 22	AD 21	AD 20	AD 19	AD 18	AD 17	AD 16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AD 15	AD 14	AD 13	AD 12	AD 11	AD 10	AD 9	AD 8	AD 7	AD 6	AD 5	AD 4	AD 3	AD 2	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 178. Address Register (ADR)

Table 186. ADR field descriptions

Field	Description
	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>

Table 186. ADR field descriptions (continued)

Field	Description
AD[22:2]	Address 20–3 AD22-2: Address 22-2 (Read Only) The Address Register provides the first failing address in the event of ECC error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address Register provides also the first address at which a ECC single error correction occurs (MCR.EDC set), if the SoC is configured to show this feature. The ECC double error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC single error correction. When accessed ADR will provide the address related to the first event occurred with the highest priority. The priorities between these 4 possible events is summarized in the following table. In User Mode the Address Register is read only.

Table 187. ADR content: priority list

Priority level	Error flag	ADR content
1	MCR[EER] = 1	Address of first ECC Double Error
2	MCR[RWE] = 1	Address of first RWW Error
3	MCR[PEG] = 0	Address of first FPEC Error
4	MCR[EDC] = 1	Address of first ECC Single Error Correction

18.4.5.8 User Test 0 register (UT0)

The User Test feature gives the user of the flash module the ability to perform test features on the flash. The User Test 0 register allows controlling the way in which the flash content check is done.

The UT0[MRE], UT0[MRV], UT0[AIS], UT0[EIE], and DSI[6:0] bits are not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x003C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBC	0	0	0	0	0	0	0							
W		E								DSI6	DSI5	DSI4	DSI3	DSI2	DSI1	DSI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0							
W										X	MRE	MRV	EIE	AIS	AIE	AID
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 179. User Test 0 register (UT0)

Table 188. UT0 field descriptions

Field	Description
UTE	<p>UTE: User Test Enable (Read/Clear)</p> <p>This status bit gives indication when User Test is enabled. All bits in UT0-1 and UMISR0-1 are locked when this bit is 0.</p> <p>This bit is not writeable to a 1, but may be cleared. The reset value is 0.</p> <p>The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write.</p> <p>For UTE the password 0xF9F99999 must be written to the UT0 register.</p>
1:8	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
DSI6-0	<p>DSI6-0: Data Syndrome Input 6-0 (Read/Write)</p> <p>These bits represents the input of Syndrome bits of ECC logic used in the ECC Logic Check. The DSI6-0 correspond to the 7 syndrome bits on a single word.</p> <p>These bits are not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect.</p> <p>0: The syndrome bit is forced at 0. 1: The syndrome bit is forced at 1.</p>
16:24	<p><i>Reserved (Read Only)</i></p> <p>A write to these bits has no effect. A read of these bits always outputs 0.</p>
25	<p><i>Reserved (Read/Write)</i></p> <p>This bit can be written and its value can be read back, but there is no function associated.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p>
MRE	<p>Margin Read Enable</p> <p>MRE enables margin reads to be done. This bit, combined with MRV, enables regular user mode reads to be replaced by margin reads.</p> <p>Margin reads are only active during Array Integrity Checks; Normal user reads are not affected by MRE.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 Margin reads are disabled. All reads are User mode reads. 1 Margin reads are enabled.</p>
MRV	<p>Margin Read Value</p> <p>If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV = 1) or to a programmed level (MRV = 0).</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 Zero's (programmed) margin reads are requested (if MRE = 1). 1 One's (erased) margin reads are requested (if MRE = 1).</p>
EIE	<p>ECC data Input Enable</p> <p>EIE enables the ECC Logic Check operation to be done.</p> <p>This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect.</p> <p>0 ECC Logic Check is disabled. 1 ECC Logic Check is enabled.</p>

Table 188. UT0 field descriptions (continued)

Field	Description
AIS	Array Integrity Sequence AIS determines the address sequence to be used during array integrity checks or Margin Mode. The default sequence (AIS = 0) is meant to replicate sequences normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is just logically sequential. It should be noted that the time to run a sequential sequence is significantly shorter than the time to run the proprietary sequence. This bit is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data, and writes have no effect. In Margin Mode only the linear sequence (AIS = 1) is allowed, while the proprietary sequence (AIS = 0) is forbidden. 0 Array Integrity sequence is a proprietary sequence. 1 Array Integrity or Margin Mode sequence is sequential.
AIE	Array Integrity Enable AIE set to 1 starts the Array Integrity Check done on all selected and unlocked blocks. The pattern is selected by AIS, and the MISR (UMISR0–4) can be checked after the operation is complete, to determine if a correct signature is obtained. AIE can be set only if MCR[ERS], MCR[PGM], and MCR[EHV] are all low. 0 Array Integrity Checks are disabled. 1 Array Integrity Checks are enabled.
AID	Array Integrity Done AID is cleared upon an Array Integrity Check being enabled (to signify the operation is on-going). Once completed, AID is set to indicate that the Array Integrity Check is complete. At this time, the MISR (UMISR0–4) can be checked. 0 Array Integrity Check is on-going. 1 Array Integrity Check is done.

18.4.5.9 User Test 1 register (UT1)

The User Test 1 register allows to enable the checks on the ECC logic related to the 32 LSB of the Double Word.

The User Test 1 register is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0040

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI
W	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI	DAI
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 180. User Test 1 register (UT1)

Table 189. UT1 field descriptions

Field	Description
DAI[31:0]	Data Array Input 31–0 These bits represent the input of the even word of ECC logic used in the ECC Logic Check. The DAI[31:0] bits correspond to the 32 array bits representing Word 0 within the double word. 0 The array bit is forced at 0. 1 The array bit is forced at 1.

18.4.5.10 User Multiple Input Signature Register 0 (UMISR0)

The Multiple Input Signature Register 0 (UMISR0) provides a mean to evaluate the array integrity. UMISR0 represents the bits 31:0 of the whole 144-bit word (2 double words including ECC).

UMISR0 is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x0048

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	031	030	029	028	027	026	025	024	023	022	021	020	019	018	017	016
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	015	014	013	012	011	010	009	008	007	006	005	004	003	002	001	000
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 181. User Multiple Input Signature Register 0 (UMISR0)

Table 190. UMSIR0 field descriptions

Field	Description
MS[031:000]	Multiple input Signature 031–000 These bits represent the MISR value obtained by accumulating the bits 31:0 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR0 register.

18.4.5.11 User Multiple Input Signature Register 1 (UMISR1)

The Multiple Input Signature Register 1 (UMISR1) provides a means to evaluate the array integrity. UMISR1 represents bits 63:32 of the whole 144-bit word (2 double words including ECC).

UMISR1 is not accessible whenever MCR[DONE] or UT0[AID] are low. Reads return indeterminate data. Writes have no effect.

Address: Base + 0x004C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	063	062	061	060	059	058	057	056	055	054	053	052	051	050	049	048
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS	MS
W	047	046	045	044	043	042	041	040	039	038	037	036	035	034	033	032
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 182. User Multiple Input Signature Register 1 (UMISR1)
Table 191. UMISR1 field descriptions

Field	Description
MS[063:032]	Multiple input Signature 063–032 These bits represent the MISR value obtained accumulating the bits 63:32 of all the pages read from the flash memory. The MS can be seeded to any value by writing the UMISR1 register.

18.4.6 Programming considerations

18.4.6.1 Modify operation

All the Modify Operations of the Flash Module are managed through the Flash User Registers Interface. All the sectors of the Flash Module belong to the same partition (Bank), therefore when a Modify operation is active on some sectors no read access is possible on any other sector (Read-While-Modify is not supported).

During a Flash Modify Operation any attempt to read any Flash location will output invalid data and bit RWE of MCR will be automatically set. This means that the Flash Module is not fetchable when a Modify Operation is active: the Modify Operation commands must be executed from another Memory (internal Ram or external Memory). If during a Modify Operation a reset occurs, the operation is suddenly terminated and the Macrocell is reset to Read Mode. The data integrity of the Flash section where the Modify Operation has been terminated or aborted is not guaranteed: the interrupted Flash Modify Operation must be repeated. In general each Modify Operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR (PGM or ERS) or UT0 (MRE or EIE).
2. The second step is the definition of the operands: the Address and the Data for programming or the Sectors for erase or margin read.
3. The third instruction is used to start the Modify Operation, by setting EHV in MCR or AIE in UT0. Once selected, but not yet started, one operation can be canceled by resetting the operation selection bit.

A summary of the available Flash modify operations is shown in the [Table 192](#).

Table 192. Flash modify operations

Operation	Select bit	Operands	Start bit
Double word program	MCR.PGM	Address and data by interlock writes	MCR.EHV
Sector erase	MCR.ERS	LMS	MCR.EHV
Array integrity check	None	LMS	UT0.AIE
Margin read	UT0.MRE	UT0.MRV + LMS	UT0.AIE
ECC logic check	UT0.EIE	UT0.DSI, UT1, UT2	UT0.AIE

Once bit MCR.EHV (or UT0.AIE) is set, all the operands can no more be modified until bit MCR.DONE (or UT0.AID) is high.

In general each modify operation is completed through a sequence of four steps:

1. Wait for operation completion: wait for bit MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UMISR0-1 with expected value).
3. Switch off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE/EIE).

In the following all the possible modify operations are described and some examples of the sequences needed to activate them are presented.

18.4.6.2 Word program

A Flash program sequence operates on any word within the Flash core.

Whenever flash bits are programmed, ECC bits also get programmed, unless the selected address belongs to a sector in which the ECC has been disabled in order to allow bit manipulation. ECC is handled on a 32-bit boundary.

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1. Addresses in locked/disabled blocks cannot be programmed.

The user may program the values in any words within a single program sequence.

The Program operation consists of the following sequence of events:

1. Change the value in the MCR.PGM bit from 0 to 1.
2. Ensure the block that contains the address to be programmed is unlocked.
 - a) Write the first address to be programmed with the program data.
 - b) The Flash module latches address bits (22:2) at this time.
 - c) The Flash module latches data written as well.
 - d) This write is referred to as a program data interlock write. An interlock is at 32 bits.
3. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 8 to terminate. .

4. Wait until the MCR[DONE] bit goes high.
5. Confirm MCR[PEG]=1.
6. Write a logic 0 to the MCR[EHV] bit.
7. If more addresses are to be programmed, return to step 2.
8. Write a logic 0 to the MCR[PGM] bit to terminate the program operation.

Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. The first write after a program is initiated determines the page address to be programmed. This first write is referred to as an interlock write. The interlock write determines if the test or normal array space will be programmed by causing MCR[PEAS] to be set/cleared. An interlock write must be performed before setting MCR[EHV]. The user may terminate a program sequence by clearing MCR[PGM] prior to setting MCR[EHV]. After the interlock write, additional writes only affect the data to be programmed in the word. If multiple writes are done to the same location the data for the last write is used in programming.

While MCR[DONE] is low and MCR[EHV] is high, the user may clear EHV, resulting in a program abort. A Program abort forces the module to step 7 of the program sequence. An aborted program will result in MCR[PEG] being set low, indicating a failed operation. MCR[DONE] must be checked to know when the aborting command has completed. The data space being operated on before the abort will contain indeterminate data. This may be recovered by repeating the same program instruction or executing an erase of the affected blocks.

Example 1. Word program of data 0x55AA55AA at address 0x00AAA8

```

MCR          = 0x00000010;          /* Set PGM in MCR: Select Operation */
(0x00AAA8)   = 0x55AA55AA;          /* Latch Address and 32 LSB data */
MCR          = 0x00000011;          /* Set EHV in MCR: Operation Start */
do
{ tmp       = MCR;                  /* Loop to wait for DONE=1 */
} while ( !(tmp & 0x00000400) );    /* Read MCR */
status      = MCR & 0x00000200;     /* Check PEG flag */
MCR         = 0x00000010;          /* Reset EHV in MCR: Operation End */
MCR         = 0x00000000;          /* Reset PGM in MCR: Deselect Operation */
    
```

18.4.6.3 Sector erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks (sectors). The test block cannot be erased.

The erase sequence is fully automated within the Flash. The user only needs to select the blocks to be erased and initiate the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, no specific operation order must be assumed.

The erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from 0 to 1.
2. Select the block(s) to be erased by writing '1's to the appropriate register(s).
Note that Lock and Select are independent. If a block is selected and locked, no erase will occur.
3. Write to any address in Flash. This is referred to as an erase interlock write.

4. Write a logic 1 to the MCR.EHV bit to start the internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR.DONE bit goes high.
6. Confirm MCR.PEG=1.
7. Write a logic 0 to the MCR.EHV bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR.ERS bit to terminate the erase operation.

After setting MCR.ERS, one write, referred to as an interlock write, must be performed before MCR.EHV can be set to 1. Data words written during erase sequence interlock writes are ignored. The User may terminate the erase sequence by clearing ERS before setting EHV.

An erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low. An erase abort forces the Module to step 8 of the erase sequence.

An aborted erase will result in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed. The block(s) being operated on before the abort contain indeterminate data. This may be recovered by executing an erase on the affected blocks. The User may not abort an erase sequence while in erase suspend.

Example 2. Erase of sectors B0F1 and B0F2

```

MCR          = 0x00000004;          /* Set ERS in MCR: Select Operation */
LMS          = 0x00000006;          /* Set LSL2-1 in LMS: Select Sectors to erase */
(0x000000)   = 0xFFFFFFFF;         /* Latch a Flash Address with any data */
MCR          = 0x00000005;          /* Set EHV in MCR: Operation Start */
do           /* Loop to wait for DONE=1 */
{ tmp       /* Read MCR */
  = MCR;
} while ( !(tmp & 0x00000400) );
status      = MCR & 0x00000200;     /* Check PEG flag */
MCR         = 0x00000004;          /* Reset EHV in MCR: Operation End */
MCR         = 0x00000000;          /* Reset ERS in MCR: Deselect Operation */

```

18.4.6.3.1 Erase suspend/resume

The erase sequence may be suspended to allow read access to the Flash Core. It is not possible to program or to erase during an erase suspend. During erase suspend, all reads to blocks targeted for erase return indeterminate data. An erase suspend can be initiated by changing the value of the MCR.ESUS bit from 0 to 1. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0 to 1 transition of MCR.ESUS causes the Module to start the sequence which places it in erase suspend.

The User must wait until MCR.DONE=1 before the Module is suspended and further actions are attempted. MCR.DONE will go high no more than tESUS after MCR.ESUS is set to 1. Once suspended, the array may be read. Flash Core reads while MCR.ESUS=1 from the block(s) being erased return indeterminate data.

Example 3. Sector erase suspend

```

MCR          = 0x00000007;          /* Set ESUS in MCR: Erase Suspend */
do           /* Loop to wait for DONE=1 */

```



```
{ tmp          = MCR;          /* Read MCR */
} while ( !(tmp & 0x00000400) );
```

Notice that there is no need to clear MCR.EHV and MCR.ERS in order to perform reads during erase suspend. The erase sequence is resumed by writing a logic 0 to MCR.ESUS. MCR.EHV must be set to '1' before MCR.ESUS can be cleared to resume the operation. The module continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Example 4. Sector erase resume

```
MCR          = 0x00000005;          /* Reset ESUS in MCR: Erase Resume */
```

18.4.6.4 User Test Mode

User Test Mode is a procedure to check the integrity of the Flash Module.

Three kinds of test can be performed:

- Array Integrity Self Check
- Margin Read
- ECC Logic Check

The User Test Mode is equivalent to a Modify operation: read accesses attempted by the user during User Test Mode generates a Read-While-Write Error (RWE of MCR set).

It is not allowed to perform User Test operations on the Test and Shadow blocks.

18.4.6.4.1 Array integrity self check

Array Integrity is checked using a pre-defined address sequence (proprietary), and this operation is executed on selected and unlocked blocks. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UMISR0-1), to determine if an incorrect read, or ECC detection was noted.

The internal MISR calculator is a 32 bit register. The 32 bit data, the 7 ECC data and the single and double ECC errors of the Word are therefore captured by the MISR through 2 different read accesses at the same location. The whole check is done through 2 complete scans of the memory address space:

1. The 1st pass will scan only bits 31-0 of each word.
2. The 2nd pass will scan only the ECC bits (7) and the single and double ECC errors (1 + 1) of each word.

The 32 data bit and the 7 ECC data are sampled before the eventual ECC correction, while the single and double error flags are sampled after the ECC evaluation. Only data from existing and unlocked locations are captured by the MISR. The MISR can be seeded to any value by writing the UMISR0-1 registers.

Once command is started, Array Integrity check is run by FPEC using system clock and the number of wait states identified by address and data wait states.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.

2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in LMS. Note that Lock and Select are independent. If a block is selected and locked, no Array Integrity Check will occur.
3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Clear (or insert seed) UMISR0-1
5. Write a logic 1 to the UT0.AIE bit to start the Array Integrity Check.
6. Wait until the UT0.AID bit goes high.
7. Compare UMISR0-1 content with the expected result.
8. Write a logic 0 to the UT0.AIE bit.
9. If more blocks are to be checked, return to step 2.
10. clear UT0 writing UT0.UTE to '0'

It is recommended to leave UT0.AIS at 0 and use the proprietary address sequence that checks the read path more fully, although this sequence takes more time. While UT0.AID is low and UT0.AIE is high, the User may clear AIE, resulting in a Array Integrity Check abort.

UT0.AID must be checked to know when the aborting command has completed.

Example 5. Array integrity check of sectors B0F1 and B0F2

```

UT0          = 0xF9F99999;          /* Set UTE in UT0: Enable User Test */
LMS          = 0x00000006;          /* Set LSL2-1 in LMS: Select Sectors */
UT0          = 0x80000002;          /* Set AIE in UT0: Operation Start */
do
/* Loop to wait for AID=1 */
{ tmp        = UT0;                /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0        = UMISR0;              /* Read UMISR0 content*/
data1        = UMISR1;              /* Read UMISR1 content*/
UT0          = 0x00000000;          /* Reset UTE and AIE in UT0: Operation End */

```

18.4.6.4.2 Margin read

Margin read procedure (either Margin 0 or Margin 1), can be run on unlocked blocks in order to unbalance the Sense Amplifiers, respect to standard read conditions, so that all the read accesses reduce the margin vs '0' (UT0.MRV = '0') or vs '1' (UT0.MRV = '1'). Locked sectors are ignored by MISR calculation and ECC flagging.

The results of the margin reads can be checked comparing checksum value in UMISR0-1. Since Margin reads are done at voltages that differ than the normal read voltage, lifetime expectancy of the Flash macrocell is impacted by the execution of Margin reads.

Doing Margin reads repetitively results in degradation of the Flash Array, and shorten expected lifetime experienced at normal read levels. It is recommended the Margin reads be done on a limited basis (less than 10 times before the next chip erase).

The Margin Read Setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in LMS. Note that Lock and Select are independent. If a block is selected and locked, no Margin Read will occur.

3. Set eventually UT0.AIS bit for a sequential addressing only.
4. Change the value in the UT0.MRE bit from 0 to 1.
5. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
6. Write a logic 1 to the UT0.AIE bit to start the Margin Read Setup or skip to step 6 to terminate.
7. Wait until the UT0.AID bit goes high.
8. Compare UMISR0-1 content with the expected result.
9. Write a logic 0 to the UT0.AIE UT0.MRE and UT0.MRV bits.

It is recommended to leave UT0.AIS at 1 and use the linear address sequence and takes less time. While UT0.AID is low and UT0.AIE is high, the User may clear AIE, resulting in a Margin Mode abort. UT0.AID must be checked to know when the aborting command has completed.

Example 6. Margin read setup versus '1's

```

UT0          = 0xF9F99999;          /* Set UTE in UT0: Enable User Test */
UT0          = 0x80000020;          /* Set MRE in UT0: Select Operation */
UT0          = 0x80000030;          /* Set MRV in UT0: Select Margin versus 1's */
UT0          = 0x80000032;          /* Set AIE in UT0: Operation Start */
do
/* Loop to wait for AID=1 */
{ tmp = UT0;
/* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0        = UMISR0;             /* Read UMISR0 content*/
data1        = UMISR1;             /* Read UMISR1 content*/
UT0          = 0x00000000;          /* Reset UTE, AIE, MRE, MRV in UT0: Deselect
Operation */
    
```

18.4.6.4.3 ECC logic check

ECC logic can be checked by forcing the input of ECC logic: the 32 bits of data and the 7 bits of ECC syndrome can be individually forced and they will drive simultaneously at the same value the ECC logic of the word.

The results of the ECC Logic Check can be verified by reading the MISR value. The ECC Logic Check operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the related password in UT0.
2. Write in UT1.DAI31-0 Word Input value.
3. Write in UT0.DSI6-0 the Syndrome Input value.
4. Select the ECC Logic Check: write a logic 1 to the UT0.EIE bit.
5. Write a logic 1 to the UT0.AIE bit to start the ECC Logic Check.
6. Wait until the UT0.AID bit goes high.
7. Compare UMISR0-1 content with the expected result.
8. Write a logic 0 to the UT0.AIE bit.

Notice that when UT0.AID is low UMISR0-1, UT1 and bits MRE, MRV, EIE, AIS and DSI6-0 of UT0 are not accessible: reading returns undeterminate data and write has no effect.

Example 7. ECC logic check

```

UT0          = 0xF9F99999;          /* Set UTE in UT0: Enable User Test */
    
```

Flash Memory

```

UT1          = 0x55555555;          /* Set DAI31-0 in UT1: Word Input Data */
UT0          = 0x80380000;          /* Set DSI6-0 in UT0: Syndrome Input Data */
UT0          = 0x80380008;          /* Set EIE in UT0: Select ECC Logic Check */
UT0          = 0x8038000A;          /* Set AIE in UT0: Operation Start */
do
/* Loop to wait for AID=1 */
{ tmp        = UT0;                /* Read UT0 */
} while ( !(tmp & 0x00000001) );
data0        = UMISR0;              /* Read UMISR0 content (expected 0x55555555) */
UT0          = 0x00000000;          /* Reset UTE, AIE and EIE in UT0: Operation End */

```

18.4.7 Error correction code

The Flash Macrocell provides a method to improve the reliability of the data stored in Flash: the usage of an Error Correction Code. ECC circuitry provides correction of single bit faults and is used to achieve automotive reliability targets. Some units will experience single bit corrections throughout the life of the product with no impact to product reliability. Word size is fixed at 32 bits.

At each Word of 32 bits there are associated 7 ECC bits that are programmed in such a way to guarantee a Single Error Correction and a Double Error Detection (SEC-DED).

18.4.7.1 ECC algorithms

The Flash module supports one ECC Algorithm: “All ‘1’s No Error”. A modified Hamming code is used that ensures the all erased state (that is, 0xFFFF....FFFF) data is a valid state, and will not cause an ECC error. This allows the user to perform a blank check after a sector erase operation.

18.4.7.2 ECC Algorithms Features

The Flash Macrocell ECC Algorithm supports the following features:

- All ‘0’s Error
 - The All ‘0’s Error Algorithm detects as Double ECC Error any Word in which all the 39 bits are “0’s).
- All ‘1’s No Error
 - The All ‘1’s No Error Algorithm detects as valid any Word read on a just erased sector (all the 39 bits are “1’s).
 - This option allows to perform a Blank Check after a Sector Erase operation.
- Bit Manipulation
 - 8 bits clears (by byte) are allowed on any erased word mantaining valid the syndrome of the word. 8 bits clears can be done on any byte of the word without a specific order. This featured is intended as a counter for EE-Emulation.

Example 1: data patterns with the same ECC syndrome (equal to 0x7F).

```

0xFFFFFFFF -> 7F
0xFFFFFFFF00 -> 7F
0xFFFF00FF -> 7F

```

```

0xFF00FFFF -> 7F
0x00FFFFFF -> 7F
0xFFFF0000 -> 7F
0x0000FFFF -> 7F
0xFF000000 -> 7F
0x000000FF -> 7F
0x00000000 -> 7F
    
```

- **Enhanced flagging**

In case flagging method is required for more than 4 writes, the following sequence allows up to 7 pattern with the same ECC syndrome.

```

0xFFFFFFFF -> 7F
0xFFFFFFFFB1 -> 7F
0xFFFFFFFF00 -> 7F
0xFFACFF00 -> 7F
0xFF00FF00 -> 7F
0xCA00FF00 -> 7F
0x0000FF00 -> 7F
0x00000000 -> 7F
    
```

- **3 Bits Error Detection**

- 40.21% of the possible 3 bits errors are detected as Double ECC Error.
- 59.79% of the possible 3 bits errors are instead detected as Single ECC Error and miscorrected..

18.4.8 Protection strategy

Two kinds of protection are available: Modify Protection to avoid unwanted program/erase in Flash sectors. The Censored Mode to avoid piracy must be managed by the associated Code Flash Macrocell embedded in the same device.

18.4.8.1 Modify protection

The Flash Modify Protection information is stored in non-volatile Flash cells located in the TestFlash. This information is read once during the Flash initialization phase following the exiting from Reset and they are stored in volatile registers that act as actuators.

The reset state of all the Volatile Modify Protection Registers is the protected state.

All the non-volatile Modify Protection registers can be programmed through a normal Word Program operation at the related locations in TestFlash.

The non-volatile Modify Protection registers cannot be erased.

- The non-volatile Modify Protection Registers are physically located in TestFlash their bits can be programmed to '0' only once and they can no more be restored to '1'.
- The Volatile Modify Protection Registers are Read/Write registers which bits can be written at '0' or '1' by the user application.

A software mechanism is provided to independently lock/unlock each Low, Mid Address Space Block against program and erase.

Flash Memory

Software locking is done through the LML (Low/Mid Address Space Block Lock Register). An alternate means to enable software locking for blocks of Low Address Space only is through the SLL (Secondary Low/Mid Address Space Block Lock Register).

All these registers have a Non Volatile image stored in TestFlash (NVLML, NVSLL), so that the locking information is kept on reset.

On delivery the TestFlash Non Volatile image is at all 1's that means all sectors locked. By programming the Non Volatile locations in TestFlash the selected sectors can be unlocked. Being the TestFlash One Time Programmable (i.e. not erasable), once unlocked the sectors cannot be locked again.

Of course, on the contrary, all the volatile registers can be written at 0 or 1 at any time, therefore the User Application can lock and unlock sectors when desired.

Chapter 19

Enhanced Direct Memory Access (eDMA)

19.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors (TCD) for each channel.

The following figure is a block diagram of the eDMA module.

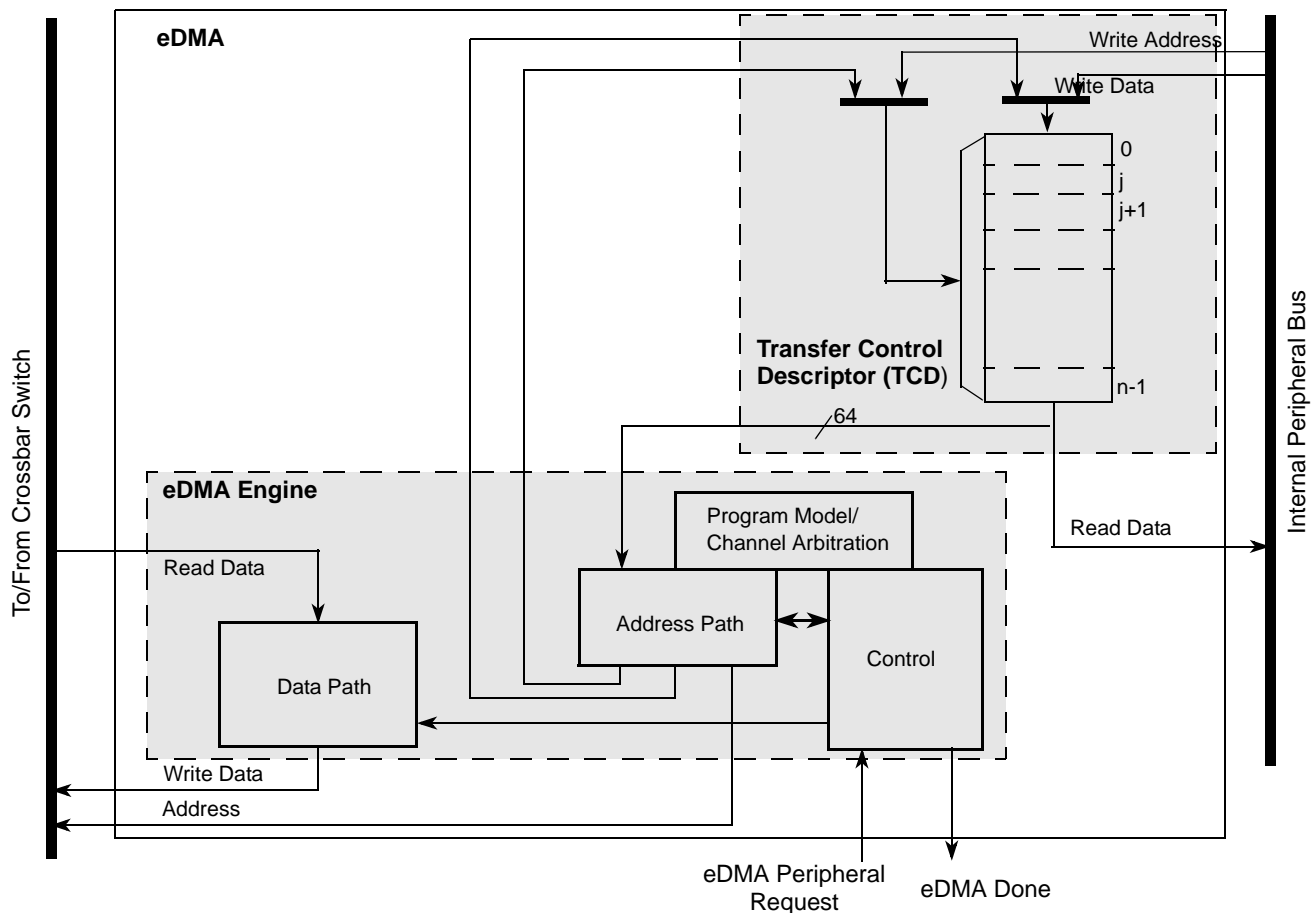


Figure 183. eDMA block diagram

19.1.1 Overview

The eDMA is a highly-programmable data-transfer engine optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the data packet itself.

Throughout this document, n is used to reference the channel number. Additionally, data sizes are defined as byte (8-bit), halfword (16-bit), word (32-bit) and doubleword (64-bit).

19.1.2 Features

The DMA module supports the following features:

- All data movement via dual-address transfers: read from source, write to destination
 - Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes
- 16-channel implementation that performs complex data transfers with minimal intervention from a host processor
 - Internal data buffer, used as temporary storage to support 16-byte burst transfers
 - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor organized to support two-deep, nested transfer operations
 - An *inner* data transfer loop defined by a “minor” byte transfer count
 - An *outer* data transfer loop defined by a “major” iteration count
- Channel service request via one of three methods:
 - Explicit software initiation
 - Initiation via a channel-to-channel linking mechanism for continuous transfers
 - Independent channel linking at end of minor loop and/or major loop
 - Peripheral-paced hardware requests (one per channel)
 - For all three methods, *one service request per execution of the minor loop is required*
- Support for fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests (chip dependent)
 - One interrupt per channel, optionally asserted at completion of major iteration count
 - Error terminations are optionally enabled per channel, and logically summed together to form a small number of error interrupt outputs
- Support for scatter/gather DMA processing

The structure of the transfer control descriptor is fundamental to the operation of the DMA module. It is defined below in a ‘C’ pseudo-code specification, where `int` refers to a 32-bit variable (unless noted otherwise) and `short` is a 16-bit variable.

NOTE

To compile these structures, change any periods ‘.’ in the variable name to underscores ‘_’.

```
typedef union {
  struct {
    unsigned short citer.linkch:6; /* citer.e_link = 1 */
    unsigned short citer:9; /* link channel number, */
  } minor_link_enabled; /* current ("major") iteration count */
  struct { /* channel link at end of the minor loop */
    unsigned short citer:15; /* citer.e_link = 0 */
  } /* current ("major") iteration count */
}
```



```

    } minor_link_disabled;          /* no linking at end of the minor loop */
} t_minor_link_citer;

typedef union {
    struct {
        unsigned short biter.linkch:6; /* biter.e_link = 1 */
        unsigned short biter:9;        /* link channel number, */
    } init_minor_link_enabled;        /* beginning ("major") iteration count */
    struct {
        unsigned short biter:15;       /* channel link at end of the minor loop */
    } init_minor_link_disabled;      /* biter.e_link = 0 */
} t_minor_link_biter;                /* beginning ("major") iteration count */
                                     /* no linking at end of the minor loop */

typedef struct {
    unsigned intsaddr;                /* source address */
    unsigned intsmod:5;                /* source address modulo */
    unsigned intssize:3;               /* source transfer size */
    unsigned intdmod:5;                /* destination address modulo */
    unsigned intdsize:3;               /* destination transfer size */
    short soff;                        /* signed source address offset */
    unsigned intnbytes;                /* inner ("minor") byte count */
    int slast;                          /* last source address adjustment */
    unsigned intdaddr;                 /* destination address */
    unsigned shortciter.e_link:1;     /* enable channel linking on minor loop */
    t_minor_link_citerminor_link_citer; /* conditional current iteration count */
    short doff;                         /* signed destination address offset */
    int dlast_sga;                     /* last destination address adjustment, or
                                     scatter/gather address (if e_sg = 1) */
    unsigned shortbiter.e_link:1;     /* beginning channel link enable */
    t_minor_link_biterminor_link_biter; /* beginning ("major") iteration count */
    unsigned intbwc:2;                 /* bandwidth control */
    unsigned intmajor.linkch:6;        /* link channel number */
    unsigned intdone:1;                 /* channel done */
    unsigned intactive:1;               /* channel executing */
    unsigned intmajor.e_link:1;        /* enable channel linking on major loop */
    unsigned inte_sg:1;                 /* enable scatter/gather descriptor */
    unsigned intd_req:1;                /* disable ipd_req when done */
    unsigned intint_half:1;            /* interrupt on citer = (biter >> 1) */
    unsigned intint_maj:1;              /* interrupt on major loop completion */
    unsigned intstart:1;                /* explicit channel start */
} tcd                                 /* transfer_control_descriptor */

```

The basic operation of a channel is defined as:

1. The channel is initialized by software loading the transfer control descriptor into the eDMA's programming model, memory-mapped through the slave bus space, and implemented as local memory.
2. The channel requests service; either explicitly by software, a peripheral request or a linkage from another channel.

NOTE

The major loop executes one iteration per service request.

3. The contents of the transfer control descriptor for the activated channel is read from the local memory and loaded into the eDMA engine's internal register file.

4. The eDMA engine executes the data transfer defined by the transfer control descriptor, reading from the source and writing to the destination. The number of iterations in the minor loop is automatically calculated by the eDMA engine. The number of iterations within the minor loop is a function of the number of bytes to transfer (nbytes), the source size (ssize) and the destination size (dsize). The completion of the minor loop is equal to one iteration of the major loop.

5. At the conclusion of the minor loop's execution, certain fields of the transfer control descriptor are written back to the local TCD memory.

The process (steps 2-5) is repeated until the outer major loop's iteration count is exhausted. At that time, additional processing steps are completed, e.g., the optional assertion of an interrupt request signaling the transfer's completion, final adjustments to the source and destination addresses, etc. A more detailed description of the channel processing is listed in the pseudo-code below. This simplified example is intended to represent basic data transfers. Detailed processing associated with the error handling is omitted.

```
/* the given eDMA channel is requesting service by the software assertion of the
   tcd[channel].start bit, the assertion of an enabled ipd_req from a device, or
   the implicit assertion of a channel-to-channel link */
```

```
/* begin by reading the transfer control descriptor from the local RAM
   into the local dma_engine registers */
```

```
dma_engine      = read_from_local_memory [channel];
dma_engine.active = 1;                    /* set active flag */
dma_engine.done  = 0;                    /* clear done flag */
```

```
/* check the transfer control descriptor for consistency */
```

```
if (dma_engine.config_error == 0) {
```

```
    /* begin execution of the inner "minor" loop transfers */
    {
```

```
        /* convert the source transfer size into a byte count */
```

```
        switch (dma_engine.ssize) {
        case 0:                                /* 8-bit transfer */
            src_xfr_size = 1;
            break;
        case 1:                                /* 16-bit transfer */
            src_xfr_size = 2;
            break;
        case 2:                                /* 32-bit transfer */
            src_xfr_size = 4;
            break;
        case 3:                                /* 16-byte burst transfer */
            src_xfr_size = 16;
            break;
        case 4:                                /* 32-byte burst transfer */
            src_xfr_size = 32;
            break;
        }
    }
```

```
        /* convert the destination transfer size into a byte count */
```

```
        switch (dma_engine.dsize) {
        case 0:                                /* 8-bit transfer */
            dest_xfr_size = 1;
```

```

        break;
    case 1:                                /* 16-bit transfer */
        dest_xfer_size = 2;
        break;
    case 2:                                /* 32-bit transfer */
        dest_xfer_size = 4;
        break;
    case 3:                                /* 64-bit transfer */
        dest_xfer_size = 8;
        break;
    case 4:                                /* 16-byte burst transfer */
        dest_xfer_size = 16;
        break;
    case 5:                                /* 32-byte burst transfer */
        dest_xfer_size = 32;
        break;
}

/* determine the larger of the two transfer sizes, this value reflects */
/* the number of bytes transferred per read->write sequence. */
/* number of iterations of the minor loop = nbytes / xfer_size */
if (dma_engine.ssize < dma_engine.dsize)
    xfer_size = dest_xfer_size;
else
    xfer_size = src_xfer_size;

/* process the source address, READ data into the buffer*/

/* read "xfer_size" bytes from the source */
/* if the ssize < dsize, do multiple reads to equal the dsize */
/* if the ssize => dsize, do a single read of source data */
number_of_source_reads = xfer_size / src_xfer_size;

for (number_of_source_reads) {
    dma_engine.data = read_from_amba-ahb (dma_engine.saddr, src_xfer_size);

    /* generate the next-state source address */
    /* sum the current saddr with the signed source offset */
    ns_addr = dma_engine.saddr + (int) dma_engine.soff; }

    /* if enabled, apply the power-of-2 modulo to the next-state addr */
    if (dma_engine.smod != 0)
        address_select = (1 << dma_engine.smod) - 1; }
    else
        address_select = 0xffff_ffff;

    dma_engine.saddr = ns_addr          & address_select
                    | dma_engine.saddr & ~address_select; }
}

/* process the destination address, WRITE data from buffer */

/* write "xfer_size" bytes to the destination */
/* if the dsize < ssize, do multiple writes to equal the ssize */
/* if the dsize => ssize, do a single write of dest data */
number_of_dest_writes = xfer_size / dest_xfer_size;

```

```

for (number_of_dest_writes) {
    write_to_amba-ahb (dma_engine.daddr, dest_xfr_size) = dma_engine.data;

    /* generate the next-state destination address */
    /* sum the current daddr with the signed destination offset */
    ns_addr = dma_engine.daddr + (int) dma_engine.doff;

    /* if enabled, apply the power-of-2 modulo to the next-state dest addr */
    if (dma_engine.dmod != 0)
        address_select = (1 << dma_engine.dmod) - 1;
    else
        address_select = 0xffff_ffff;

    dma_engine.daddr = ns_addr          & address_select
                    | dma_engine.daddr & ~address_select;
}
if (cancel_transfer)
    break;

/* check for a higher priority channel to service if: */
/* 1) preemption is enabled */
/* 2) in fixed arbitration mode */
/* 3) a higher priority channel is requesting service */
/* 4) not already servicing a preempting channel */
if ((DCHPRIn.ecp = 1) & fixed_arbitration_mode
    higher_pri_request & ~current_channel_is_preempt)
    service_preempt_channel;

/* the bandwidth control field determines when the next read/write occurs */
if (dma_engine.bwc > 1)
    stall_dma_engine (1 << dma_engine.bwc);

/* decrement the minor loop byte count */
dma_engine.nbytes = dma_engine.nbytes - xfr_size;
}while (dma_engine.nbytes > 0) /* end of minor inner loop */

dma_engine.citer--;          /* decrement major loop iteration count */

/* if the major loop is not yet exhausted, update certain TCD values in the RAM */
if (dma_engine.citer != 0) {
    write_to_local_memory [channel].saddr = dma_engine.saddr;
    write_to_local_memory [channel].daddr = dma_engine.daddr;
    write_to_local_memory [channel].citer = dma_engine.citer;

    /* if minor loop linking is enabled, make the channel link */
    if (dma_engine.citer.e_link)
        TCD[citer.linkch].start = 1;    /* specified channel service req */

    /* check for interrupt assertion if half of the major iterations are done */
    if (dma_engine.int_half && (dma_engine.citer == (dma_engine.biter >> 1)))
        generate_interrupt (channel);

    dma_engine.active = 0;          /* clear the channel busy flag */
}

```

```

else { /* major loop is complete, dma_engine.citer == 0 */
    /* since the major loop is complete, perform the final address adjustments */

    /* sum the current {src,dst} addresses with "last" adjustment */
    write_to_local_memory [channel].saddr = dma_engine.saddr + dma_engine.slast;
    write_to_local_memory [channel].daddr = dma_engine.daddr + dma_engine.dlast;
    /* restore the major iteration count to the beginning value */
    write_to_local_memory [channel].citer = dma_engine.biter;

    /* check for interrupt assertion at completion of the major iteration */
    if (dma_engine.int_maj)
        generate_interrupt (channel);

    /* check if the ipd_req is to be disabled at completion of the major iteration */
    if (dma_engine.d_req)
        DMAERQ [channel] = 0;

    /* check for a scatter/gather transfer control descriptor */
    if (dma_engine.e_sg) {
        /* load new transfer control descriptor from the address defined by dlast_sga */
        write_to_local_memory [channel] =
            read_from_amba-ahb(dma_engine.dlast_sga,32);
    }
    if (dma_engine.major.e_link)
        TCD[major.linkch].start = 1;          /* specified channel service req */

    dma_engine.active = 0;                    /* clear the channel busy flag */
    dma_engine.done = 1;                     /* set the channel done flag */
}
else { /* configuration error detected, abort the channel */
    dma_engine.error_status = error_type; /* record the error */
    dma_engine.active = 0;                 /* clear the channel busy flag */
    /* check for interrupt assertion on error */
    if (dma_engine.int_err)
        generate_interrupt (channel);
}
}

```

For more details, consult [Section 19.2.1, Register descriptions](#) and [Section 19.3, Functional description](#).

19.2 Memory map/register definition

The eDMA's programming model is partitioned into two sections, both mapped into the slave bus space: the first region defines a number of registers providing control functions, while the second region corresponds to the local transfer control descriptor memory.

Reading an *unimplemented* register bit or memory location will return the value of zero. Write the value of zero to *unimplemented* register bits. Any access to a *reserved* memory location will result in a bus error. *Reserved* memory locations are indicated in the memory map. For 16- and 32-channel implementations, reserved memory also includes the high order "H" registers containing channels 63-32 data (i.e., DMAERQH, DMAEEIH, DMAINTH, DMAERRH).

Many of the control registers have a bit width that matches the number of channels implemented in the module, i.e., 16-, 32- or 64-bits in size. Registers associated with a 64-channel design are implemented as two 32-bit registers, and include an "H" and "L" suffixes, signaling the "high" and "low" portions of the

control function. The descriptions in this section define the 64-channel implementation. For 16- or 32-channel designs, the unused bits are not implemented: reads return zeroes, and writes are ignored.

The eDMA module does not include any logic which provides access control. Rather, this function is supported using the standard access control logic provided by the PBRIDGE controller.

Table 193 is a 32-bit view of the eDMA's memory map.

Table 193. eDMA 32-bit memory map

eDMA Offset	Register			
0x0000	eDMA Control Register (DMACR)			
0x0004	eDMA Error Status (DMAES)			
0x0008	Reserved			
0x000c	eDMA Enable Request Low (DMAERQL, Channels 31-00)			
0x0010	Reserved			
0x0014	eDMA Enable Error Interrupt Low (DMAEEIL, Channels 31-00)			
0x0018	eDMA Set Enable Request (DMASERQ)	eDMA Clear Enable Request (DMACERQ)	eDMA Set Enable Error Interrupt (DMASEEI)	eDMA Clear Enable Error Interrupt (DMACEEI)
0x001c	eDMA Clear Interrupt Request (DMACINT)	eDMA Clear Error (DMACERR)	eDMA Set Start Bit (DMASSRT)	eDMA Clear Done Status Bit (DMACDNE)
0x0020	Reserved			
0x0024	eDMA Interrupt Request Low (DMAINTL, Channels 31-00)			
0x0028	Reserved			
0x002c	eDMA Error Low (DMAERRL, Channels 31-00)			
0x0030	Reserved			
0x0034	eDMA Hardware Request Status Low (DMAHRSL, Channels 31-00)			
0x0038 – 0x00FF	Reserved			
0x0100	eDMA Channel 0 Priority (DCHPRI0)	eDMA Channel 1 Priority (DCHPRI1)	eDMA Channel 2 Priority (DCHPRI2)	eDMA Channel 3 Priority (DCHPRI3)
0x0104	eDMA Channel 4 Priority (DCHPRI4)	eDMA Channel 5 Priority (DCHPRI5)	eDMA Channel 6 Priority (DCHPRI6)	eDMA Channel 7 Priority (DCHPRI7)
0x0108	eDMA Channel 8 Priority (DCHPRI8)	eDMA Channel 9 Priority (DCHPRI9)	eDMA Channel 10 Priority (DCHPRI10)	eDMA Channel 11 Priority (DCHPRI11)
0x010c	eDMA Channel 12 Priority (DCHPRI12)	eDMA Channel 13 Priority (DCHPRI13)	eDMA Channel 14 Priority (DCHPRI14)	eDMA Channel 15 Priority (DCHPRI15)
0x0110-0x0fff	Reserved			
0x1000-0x11ff	TCD00-TCD15			
0x1200-0x3fff	Reserved			

19.2.1 Register descriptions

19.2.1.1 eDMA Control Register (DMACR)

The 32-bit DMACR defines the basic operating configuration of the eDMA.

Arbitration can be configured to use either a fixed-priority or a round-robin selection. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The priorities are assigned by the channel priority registers (see [Section 19.2.1.16, eDMA Channel n Priority \(DCHPRIn\), n = 0,..., {15}](#)). In round-robin arbitration mode, the channel priorities are ignored and the channels are cycled through without regard to priority.

Minor loop offsets are address offset values added to the final source address (saddr) or destination address (daddr) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (mloff) is added to the final source address (saddr), or the final destination address (daddr), or both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (slast and dlast_sga) are used to compute the next saddr and daddr values.

When minor loop mapping is enabled (DMACR[EMLM] = 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: an source enable bit (smloe) to specify the minor loop offset should be applied to the source address (saddr) upon minor loop completion, an destination enable bit (dmloe) to specify the minor loop offset should be applied to the destination address (daddr) upon minor loop completion, and the sign extended minor loop offset value (mloff). The same offset value (mloff) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (smloe set or dmloe set), the nbytes field is reduced to 10 bits. When both minor loop offsets are disabled (smloe cleared and dmloe cleared), the nbytes field is a 30-bit vector.

When minor loop mapping is disabled (DMACR[EMLM] = 0), all 32 bits of TCDn word2 are assigned to the nbytes field. See [Section 19.2.1.17, Transfer Control Descriptor \(TCD\)](#), for more details.

See [Figure 184](#) and [Table 194](#) for the DMACR definition.

Register address: DMA_Offset + 0x0000

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		0	0	0	0	0	0	0	0	0	0	0	0	0	0	CX	ECX
W																	
RESET:																0	0
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0	0	0	0	0	0	GRP0PRI	EML	CLM	HALT	HOE	ERG	ERC	EDB	EBW	
W								M					A	A	G		
RESET:								0	0	0	0	0	0	0	0	0	

= Unimplemented

Figure 184. eDMA Control Register (DMACR)

Table 194. eDMA Control Register (DMACR) field descriptions

Name	Description	Value
CX	Cancel Transfer	<p>0 Normal operation.</p> <p>1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.</p>
ECX	Error Cancel Transfer	<p>0 Normal operation.</p> <p>1 Cancel the remaining data transfer in the same fashion as the CX cancel transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel cancel has been honored. In addition to cancelling the transfer, the ECX treats the cancel as an error condition; thus updating the DMAES register and generating an optional error interrupt (see Section 19.2.1.2, eDMA Error Status (DMAES)).</p>
GRP0PRI	Channel Group 0 Priority	Group 0 priority level when fixed priority group arbitration is enabled.
EMLM	Enable Minor Loop Mapping	<p>0 Minor loop mapping disabled. TCDn.word2 is defined as a 32-bit NBYTES field.</p> <p>1 Minor loop mapping enabled. When set, TCDn.word2 is redefined to include individual enable fields, an offset field and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.</p>
CLM	Continuous Link Mode	<p>0 A minor loop channel link made to itself will go through channel arbitration before being activated again.</p> <p>1 A minor loop channel link made to itself will not go through channel arbitration before being activated again. Upon minor loop completion the channel will active again if that channel has has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.</p>
HALT	Halt eDMA Operations	<p>0 Normal operation.</p> <p>1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution will resume when the HALT bit is cleared.</p>
HOE	Halt On Error	<p>0 Normal operation.</p> <p>1 Any error will cause the HALT bit to be set. Subsequently, all service requests will be ignored until the HALT bit is cleared.</p>

Table 194. eDMA Control Register (DMACR) field descriptions

Name	Description	Value
ERGA	Enable Round Robin Group Arbitration	0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
ERCA	Enable Round Robin Channel Arbitration	0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
EDBG	Enable Debug	0 The assertion of the ipg_debug input is ignored. 1 The assertion of the ipg_debug input causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the ipg_debug input is negated or the EDBG bit is cleared.
EBW	Enable Buffered Writes	0 The bufferable write signal (hprot[2]) is not asserted during AMBA AHB writes. 1 The bufferable write signal (hprot[2]) is asserted on all AMBA AHB writes except for the last write sequence.

19.2.1.2 eDMA Error Status (DMAES)

The DMAES register provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on 0-modulo-transfer_size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively. In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled. If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (dlast_sga) is not aligned on a 32-byte boundary. If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD.citer.e_link bit does not equal the TCD.biter.e_link bit. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and assert an error interrupt request, if enabled. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address and current iteration count at the point of the fault. When a system bus error occurs, the channel is terminated after the read or write transaction which

is already pipelined after errant access, has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write will execute using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence will execute before the channel is terminated due to the destination bus error.

A transfer may be cancelled by software via the DMACR[*CX*] bit. When a cancel transfer request is recognized, the eDMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the DMAES register is updated with the cancelled channel number and error cancel bit is set. The TCD of a cancelled channel has the source address and destination address of the last transfer saved in the TCD. It is the responsibility of the user to initialize the TCD again should the channel need to be restarted because the aforementioned fields have been modified by the eDMA engine and no longer represent the original parameters. When a transfer is cancelled via the error cancel transfer mechanism (setting the DMACR[*ECX*]), the channel number is loaded into the ERRCHN field and the *ECX* and *VLD* bits are set in the DMAES register. In addition, an error interrupt may be generated if enabled. See [Section 19.2.1.14, eDMA Error \(DMAERRL\)](#) for error interrupt details. The occurrence of any type of error causes the eDMA engine to immediately stop, and the appropriate channel bit in the eDMA Error register to be asserted. At the same time, the details of the error condition are loaded into the DMAES register. The major loop complete indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are *not* affected when an error is detected. See [Figure 185](#) and [Table 195](#) for the DMAES definition.

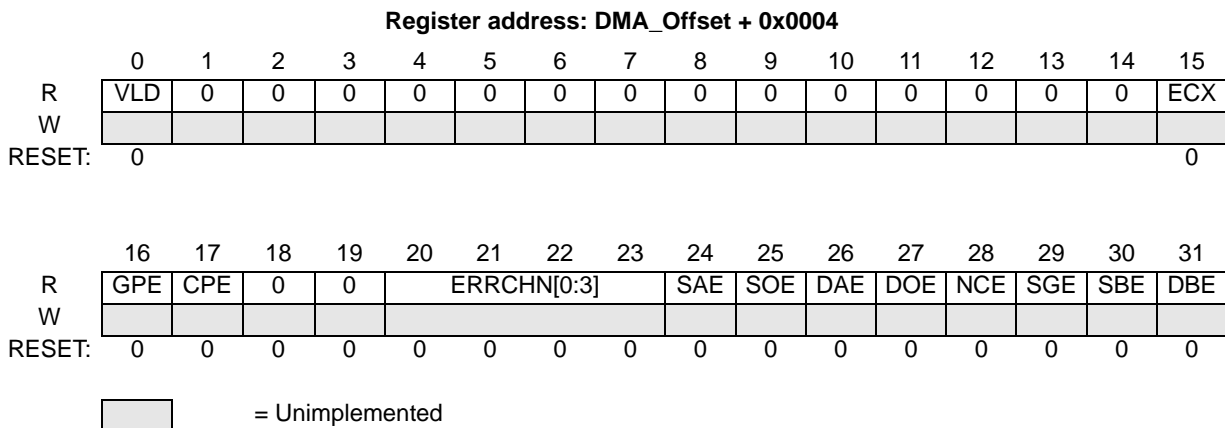


Figure 185. eDMA Error Status (DMAES) Register

Table 195. eDMA Error Status (DMAES) field descriptions

Name	Description	Value
VLD	Logical OR of all DMAERRH and DMAERRL status bits.	0 No DMAERR bits are set. 1 At least one DMAERR bit is set indicating a valid error exists that has not been cleared.
ECX	Transfer cancelled	0 No cancelled transfers. 1 The last recorded entry was a cancelled transfer via the error cancel transfer input.

Table 195. eDMA Error Status (DMAES) field descriptions (Continued)

Name	Description	Value
GPE	Group Priority Error	0 No group priority error. 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
CPE	Channel Priority Error	0 No channel priority error. 1 The last recorded error was a configuration error in the channel priorities within a group. All channel priorities within a group are not unique.
ERRCHN[0:3]	Error/Cancelled Channel Number	The channel number of the last recorded error (excluding GPE and CPE errors) or last recorded transfer that was cancelled with error exit.
SAE	Source Address Error	0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCD.saddr field. TCD.saddr is inconsistent with TCD.ssize.
SOE	Source Offset Error	0 No source offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.soff field. TCD.soff is inconsistent with TCD.ssize.
DAE	Destination Address Error	0 No destination address configuration error. 1 The last recorded error was a configuration error detected in the TCD.daddr field. TCD.daddr is inconsistent with TCD.dsize.
DOE	Destination Offset Error	0 No destination offset configuration error. 1 The last recorded error was a configuration error detected in the TCD.doff field. TCD.doff is inconsistent with TCD.dsize.
NCE	Nbytes/Citer Configuration Error	0 No nbytes/citer configuration error. 1 The last recorded error was a configuration error detected in the TCD.nbytes or TCD.citer fields. TCD.nbytes is not a multiple of TCD.ssize and TCD.dsize, or TCD.citer is equal to zero, or TCD.citer.e_link is not equal to TCD.biter.e_link.
SGE	Scatter/Gather Configuration Error	0 No scatter/gather configuration error. 1 The last recorded error was a configuration error detected in the TCD.dlast_sga field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD.e_sg is enabled. TCD.dlast_sga is not on a 32 byte boundary.
SBE	Source Bus Error	0 No source bus error. 1 The last recorded error was a bus error on a source read.
DBE	Destination Bus Error	0 No destination bus error. 1 The last recorded error was a bus error on a destination write.

19.2.1.3 eDMA Enable Request (DMAERQL)

The DMAERQL register provide a bit map for the implemented channels to enable the request signal for each channel. DMAERQL covers channels 15-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the DMASERQ and DMACERQ registers. The eDMA{S,C}ERQ registers are provided so that the request enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the DMAERQ{H,L} registers.

Both the eDMA request input signal and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does *not* affect a channel service request made explicitly through software or a linked channel request. See [Figure 186](#) and [Table 196](#) for the DMAERQ definition.

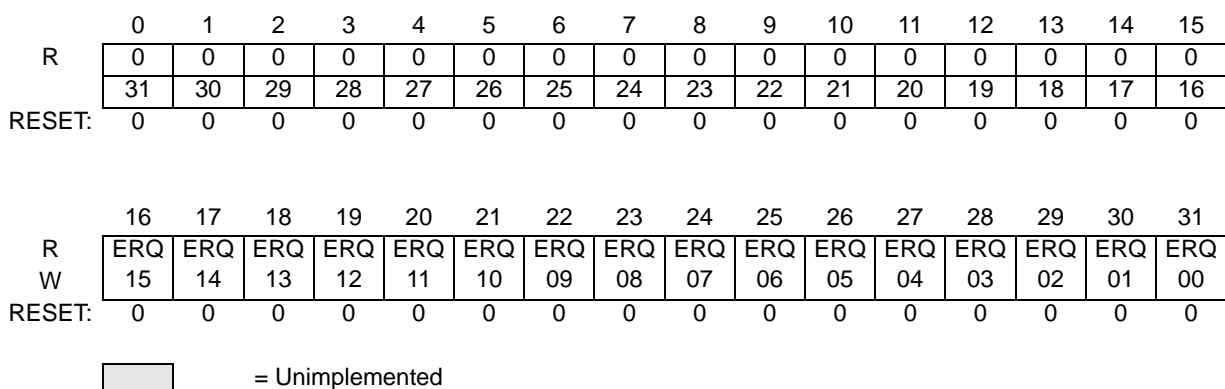


Figure 186. eDMA Enable Request (DMAERQL) Register

Table 196. eDMA Enable Request (DMAERQL) field description

Name	Description	Value
ERQn, n = 0,... 15	Enable eDMA Request n	0 The eDMA request signal for channel n is disabled. 1 The eDMA request signal for channel n is enabled.

As a given channel completes the processing of its major iteration count, there is a flag in the transfer control descriptor that may affect the ending state of the DMAERQ bit for that channel. If the TCD.d_req bit is set, then the corresponding DMAERQ bit is cleared, disabling the eDMA request; else if the d_req bit is cleared, the state of the DMAERQ bit is unaffected.

19.2.1.4 eDMA Enable Error Interrupt (DMAEEIL)

The DMAEEIL register provides a bit map for the implemented channels to enable the error interrupt signal for each channel. DMAEEIL covers channels 15-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the DMASEEI and DMACEEI registers. The eDMA{S,C}EEI registers are provided so that the error interrupt enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the DMAEEIL register.

Both the eDMA error indicator and this error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted. See [Figure 187](#) and [Table 197](#) for the DMAEEI definition.

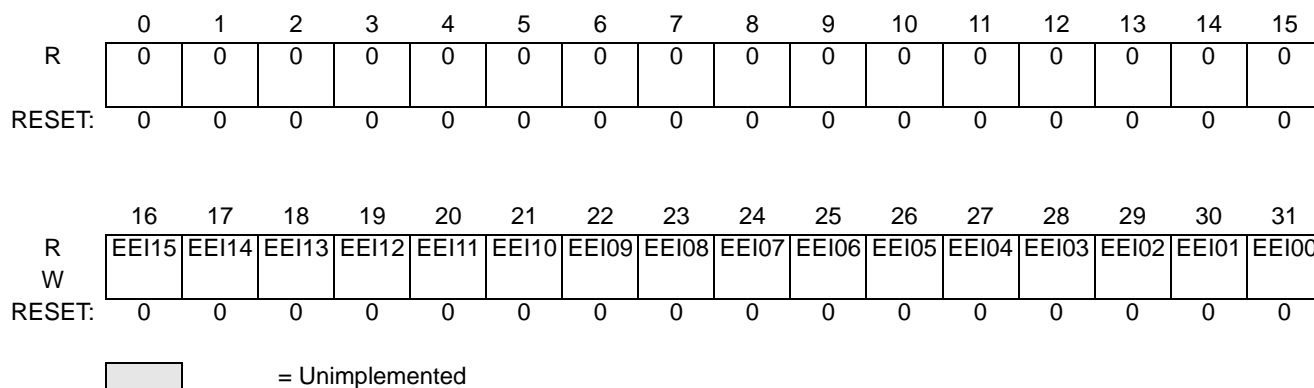


Figure 187. eDMA Enable Error Interrupt (DMAEEIL) Register

Table 197. eDMA Enable Error Interrupt (DMAEEIL) field descriptions

Name	Description	Value
EEIn, n = 0,... 15	Enable Error Interrupt n	0 The error signal for channel n does not generate an error interrupt. 1 The assertion of the error signal for channel n generate an error interrupt request.

19.2.1.5 eDMA Set Enable Request (DMASERQ)

The DMASERQ register provides a simple memory-mapped mechanism to set a given bit in the DMAERQL register to enable the eDMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQL register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAERQL to be asserted. If NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Figure 188](#) and [Table 198](#) for the DMASERQ definition.

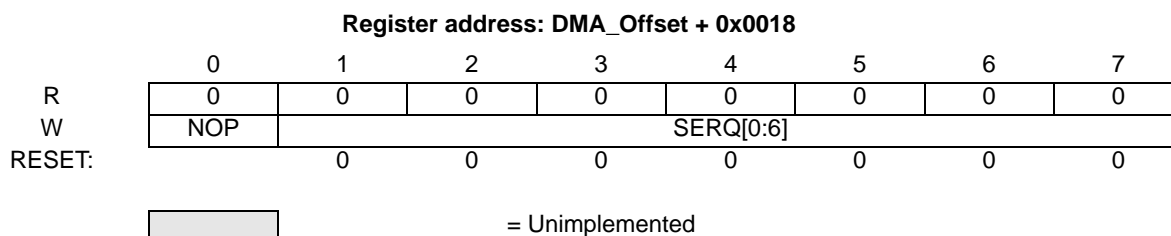


Figure 188. eDMA Set Enable Request (DMASERQ) Register

Table 198. eDMA Set Enable Request (DMASERQ) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
SERQ[0:6]	Set Enable Request	0-15 Set the corresponding bit in DMAERQL 16-63 Reserved 64-127 Set all bits in DMAERQL Note: Bit 2 (SERQ1) and Bit 3 (SERQ2) is not used.

19.2.1.6 eDMA Clear Enable Request (DMACERQ)

The DMACERQ register provides a simple memory-mapped mechanism to clear a given bit in the DMAERQL register to disable the eDMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERQL to be zeroed, disabling all eDMA request inputs. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Figure 189](#) and [Table 199](#) for the DMACERQ definition.

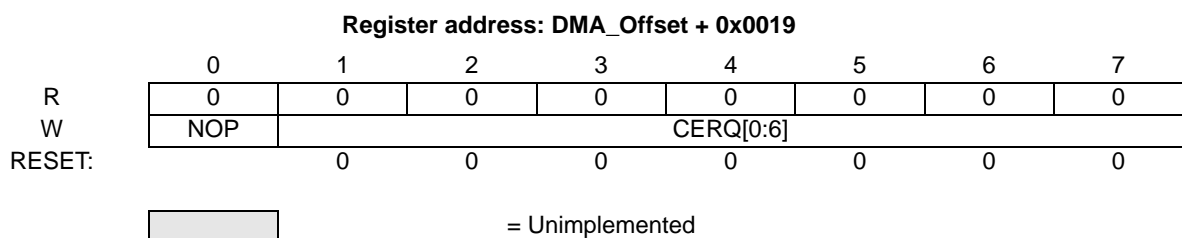


Figure 189. eDMA Clear Enable Request (DMACERQ) Register

Table 199. eDMA Clear Enable Request (DMACERQ) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
CERQ[0:6]	Clear Enable Request	0-15 Clear corresponding bit in DMAERQL 16-63 Reserved 64-127 Clear all bits in DMAERQL Note: Bit 2 (CERQ1) and Bit 3 (CERQ2) is not used.

19.2.1.7 eDMA Set Enable Error Interrupt (DMASEEI)

The DMASEEI register provides a simple memory-mapped mechanism to set a given bit in the DMAEEIL register to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEIL register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAEEI to be asserted. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Figure 190](#) and [Table 200](#) for the DMASEEI definition.

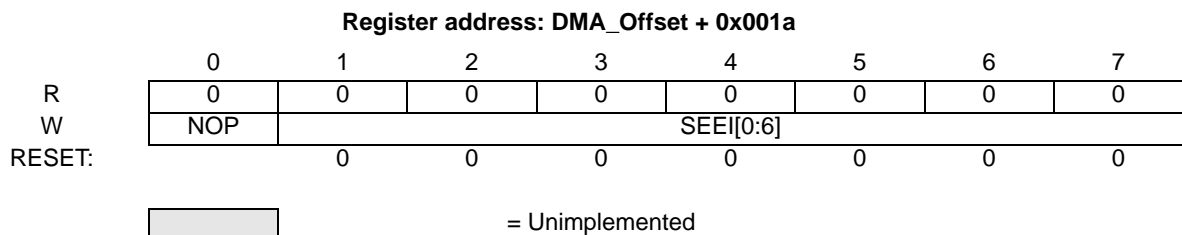


Figure 190. eDMA Set Enable Error Interrupt (DMASEEI) Register

Table 200. eDMA Set Enable Error Interrupt (DMASEEI) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
SEEI[0:6]	Set Enable Error Interrupt	0-15 Set the corresponding bit in DMAEEIL 16-63 Reserved 64-127 Set all bits in DMAEEIL Note: Bit 2 (SEEI1) and Bit 3 (SEEI2) is not used.

19.2.1.8 eDMA Clear Enable Error Interrupt (DMACEEI)

The DMACEEI register provides a simple memory-mapped mechanism to clear a given bit in the DMAEEIL register to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEIL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAEEIL to be zeroed, disabling all eDMA request inputs. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Figure 191](#) and [Table 201](#) for the DMACEEI definition.

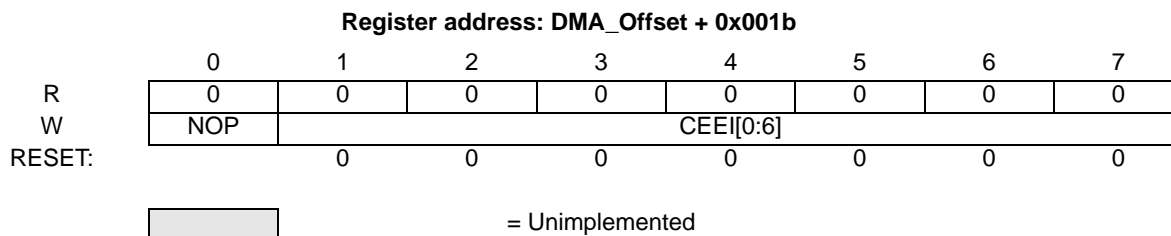


Figure 191. eDMA Clear Enable Error Interrupt (DMACEEI) Register

Table 201. eDMA Clear Enable Error Interrupt (DMACEEI) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
CEEI[0:6]	Clear Enable Error Interrupt	0-15 Clear corresponding bit in DMAEEIL 16-63 Reserved 64-127 Clear all bits in DMAEEIL Note: Bit 2 (CEEI1) and Bit 3 (CEEI2) is not used.

19.2.1.9 eDMA Clear Interrupt Request (DMACINT)

The DMACINT register provides a simple memory-mapped mechanism to clear a given bit in the DMAINTL registers to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the DMAINTL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAINTL to be zeroed, disabling all eDMA interrupt requests. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Figure 192](#) and [Table 202](#) for the DMACINT definition.

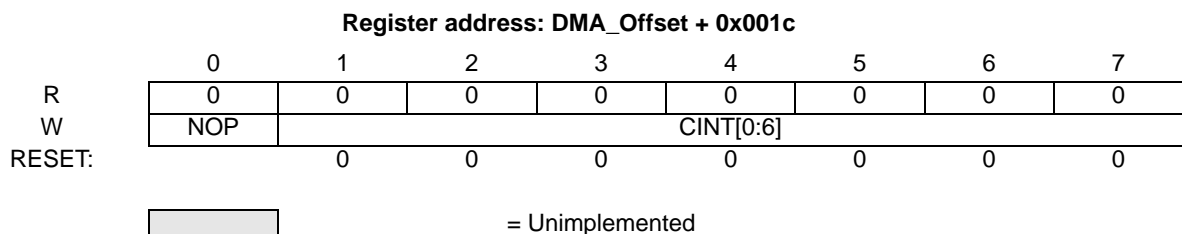


Figure 192. eDMA Clear Interrupt Request (DMACINT) Fields

Table 202. eDMA Clear Interrupt Request (DMACINT) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
CINT[0:6]	Clear Interrupt Request	0-15 Clear the corresponding bit in DMAINTL 16-63 Reserved 64-127 Clear all bits in DMAINTL Note: Bit 2 (CINT1) and Bit 3 (CINT2) is not used.

19.2.1.10 eDMA Clear Error (DMACERR)

The DMACEER register provides a simple memory-mapped mechanism to clear a given bit in the DMAERRL register to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the DMAERRL register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERRL to be zeroed, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Figure 193](#) and [Table 203](#) for the DMACERR definition.

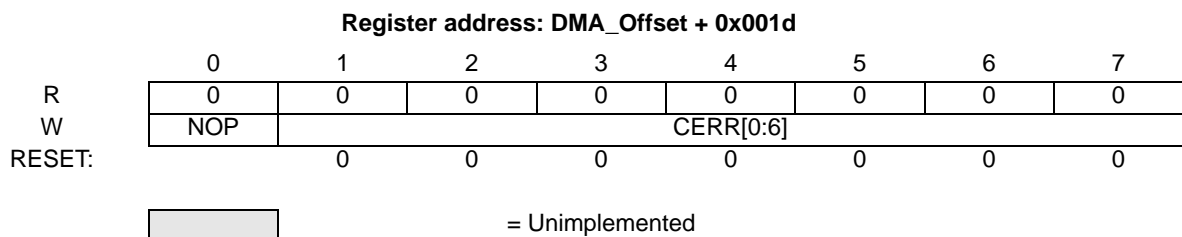


Figure 193. eDMA Clear Error (DMACERR) Register

Table 203. eDMA Clear Error (DMACERR) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
CERR[0:6]	Clear Error Indicator	0-15 Clear corresponding bit in DMAERRL 16-63 Reserved 64-127 Clear all bits in DMAERRL Note: Bit 2 (CERR1) and Bit 3 (CERR2) is not used.

19.2.1.11 eDMA Set START Bit (DMASSRT)

The DMASSRT register provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding Transfer Control Descriptor to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Table 218](#) for the TCD START bit definition.

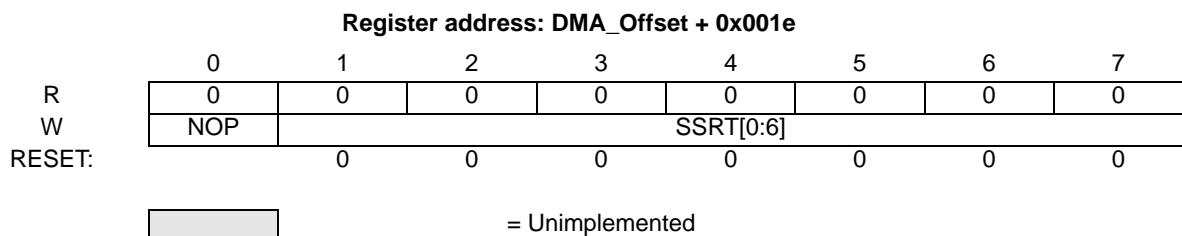


Figure 194. eDMA Set START Bit (DMASSRT) Register

Table 204. eDMA Set START Bit (DMASSRT) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
SSRT[0:6]	Set START Bit (Channel Service Request)	0-15 Set the corresponding channel's TCD start bit 16-63 Reserved 64-127 Set all TCD.start bits Note: Bit 2 (SSRT1) and Bit 3 (SSRT2) is not used.

19.2.1.12 eDMA Clear DONE Status (DMACDNE)

The DMACDNE register provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding Transfer Control Descriptor to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing all DONE bits to be cleared. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See [Table 218](#) for the TCD DONE bit definition.

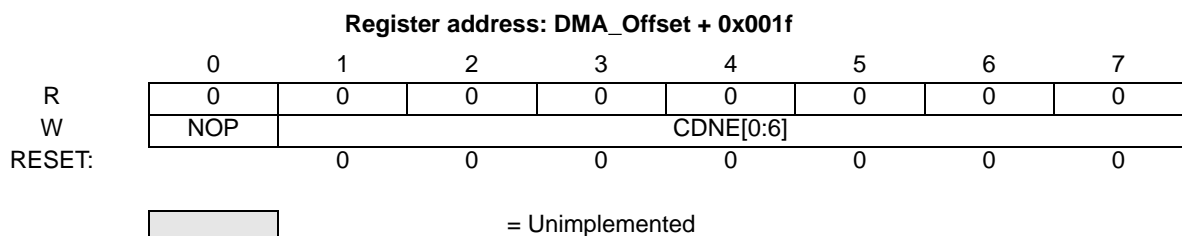


Figure 195. eDMA Clear DONE Status (DMACDNE) Register

Table 205. eDMA Clear DONE Status (DMACDNE) field descriptions

Name	Description	Value
NOP	No Operation	0 Normal operation. 1 No operation, ignore bits 1-7
CDNE[0:6]	Clear DONE Status Bit	0-15 Clear the corresponding channel's TCD Start bit 16-63 Reserved 64-127 Clear all TCD.Start bits Note: Bit 2 (CDNE1) and Bit 3 (CDNE2) is not used.

19.2.1.13 eDMA Interrupt Request (DMAINTL)

The DMAINTL register provides a bit map for the implemented channels signaling the presence of an interrupt request for each channel. DMAINTL covers channels 15-00. The eDMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any given channel, it is software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the DMACINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the DMACINT register. On writes to the DMAINT, a one in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The DMACINT register is provided so the interrupt request for a *single* channel can easily be cleared without the need to perform a read-modify-write sequence to the DMAINTL registers. See [Figure 196](#) and [Table 206](#) for the DMAINT definition.

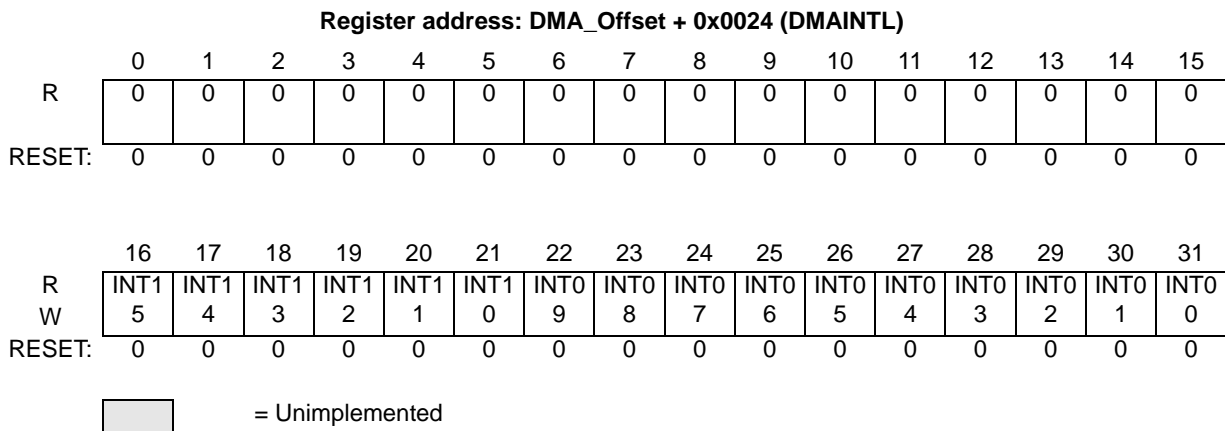


Figure 196. eDMA Interrupt Request (DMAINTL) Register

Table 206. eDMA Interrupt Request (DMAINTL) field descriptions

Name	Description	Value
INTn, n = 0,... 15	DMA Interrupt Request n	0 The interrupt request for channel n is cleared. 1 The interrupt request for channel n is active.

19.2.1.14 eDMA Error (DMAERRL)

The DMAERRL registers provide a bit map for the implemented channels signaling the presence of an error for each channel. DMAERRL covers channels 15-00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the DMAEEI register, then logically summed across groups to form several group error interrupt requests which is then routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any eDMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the DMACERR register in the interrupt service routine is used for this purpose. Recall the normal eDMA channel completion indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are *not* affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the DMAEEI register. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the DMACERR register. On writes to the DMAERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The DMACERR register is provided so the error indicator for a *single* channel can easily be cleared. See [Figure 197](#) and [Table 207](#) for the DMAERR definition.

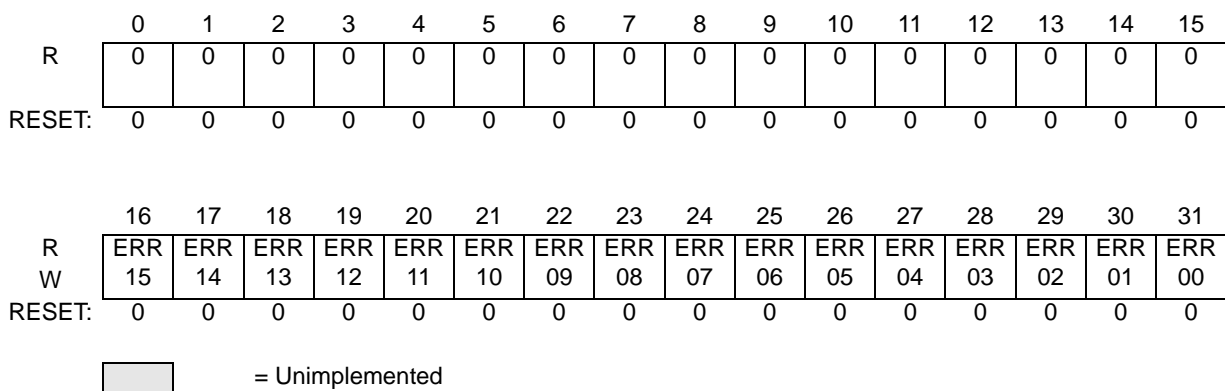


Figure 197. eDMA Error (DMAERRL) Register

Table 207. eDMA Error (DMAERRL) field descriptions

Name	Description	Value
ERRn, n = 0,... 15	DMA Error n	0 An error in channel n has not occurred. 1 An error in channel n has occurred.

19.2.1.15 eDMA Hardware Request Status (DMAHRSL)

The DMAHRSL registers provide a bit map for the implemented channels to show the current hardware request status for each channel. DMAHRSL covers channels 15-00. Hardware request status reflects the current state of the registered and qualified (via the DMAERQ field) ipd_req lines as seen by the DMA2's arbitration logic. This view into the hardware request signals may be used for debug purposes.

See [Figure 198](#) and [Figure 208](#) for the DMAHRSL definition.

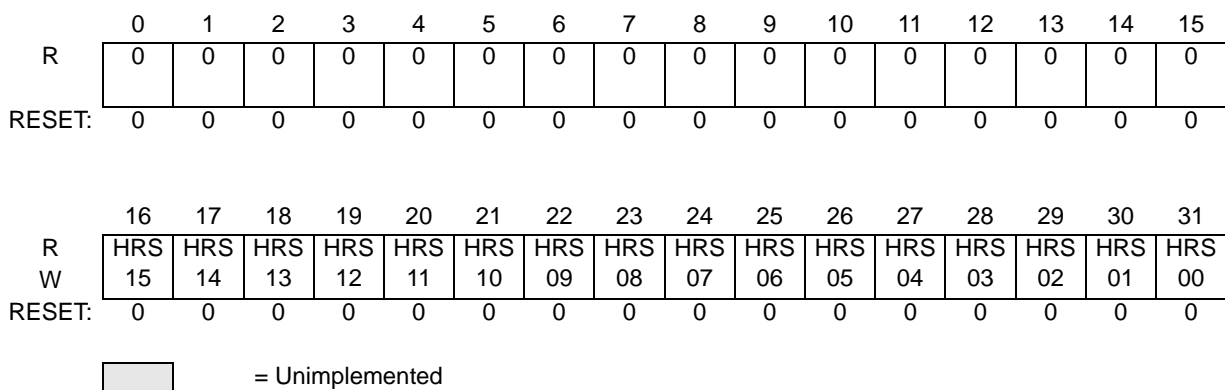


Figure 198. eDMA Hardware Request Status (DMAHRSL) Register

Table 208. eDMA Hardware Request Status (DMAHRSL) field descriptions

Name	Description	Value
HRS _n , n = 0,... 15	DMA Hardware Request Status n	0 A hardware service request for channel n is not present. 1 A hardware service request for channel n is present. Note: The hardware request status reflects the state of the request as seen by the arbitration logic. Therefore, this status is affected by the DMAERQ _n bit.

19.2.1.16 eDMA Channel n Priority (DCHPRIn), n = 0,..., {15}

When the fixed-priority channel arbitration mode is enabled (DMACR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value, i.e., 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values, otherwise a configuration error will be reported. The range of the priority value is limited to the values of 0–15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMACR. See [Figure 184](#) and [Table 194](#) for the DMACR definition.

Channel preemption is enabled on a per channel basis by setting the ECP bit in the DCHPRIn register. Channel preemption allows the executing channel’s data transfers to be temporarily suspended in favor of starting a higher priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel will be suspended and the higher priority channel will be serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes. A channel’s ability to preempt another channel can be disabled by setting the DPA bit in the DCHPRIn register. When a channel’s preempt ability is disabled, that channel cannot suspend a lower priority channel’s data transfer; regardless of the lower priority channel’s ECP setting. This allows for a pool of low priority, large data moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority

channel from consuming the preempt slot normally available a true, high priority channel. See [Figure 199](#) and [Table 209](#) for the DCHPRIn definition.

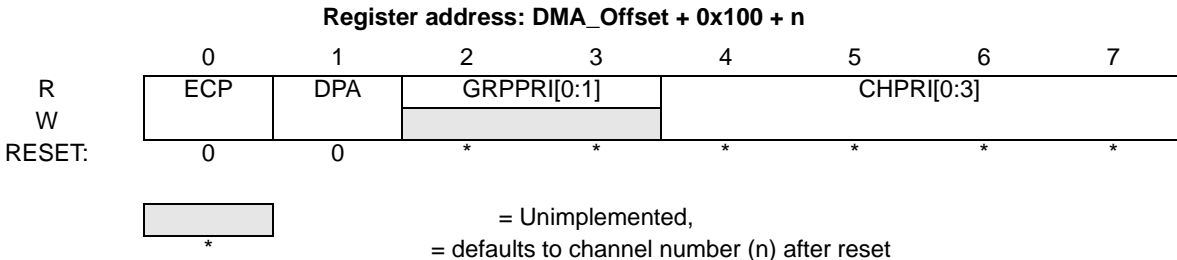


Figure 199. eDMA Channel n Priority (DCHPRIn) Register

Table 209. eDMA Channel n Priority (DCHPRIn) field descriptions

Name	Description	Value
ECP	Enable Channel Preemption	0 Channel <i>n</i> cannot be suspended by a higher priority channel's service request. 1 Channel <i>n</i> can be temporarily suspended by the service request of a higher priority channel.
DPA	Disable Preempt Ability	0 Channel <i>n</i> can suspend a lower priority channel. 1 Channel <i>n</i> cannot suspend any channel, regardless of channel priority.
GRPPRI[0:1]	Channel <i>n</i> Current Group Priority	Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored.
CHPRI[0:3]	Channel <i>n</i> Arbitration Priority	Channel priority when fixed-priority arbitration is enabled.

19.2.1.17 Transfer Control Descriptor (TCD)

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The TCD structure was previously discussed in detail in [Section 19.1.2, Features.](#) The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel [n-1]. The definitions of the TCD are presented as eight 32-bit values. [Table 210](#) is a 32-bit view of the basic TCD structure.

Table 210. TCDn 32-bit memory structure

eDMA Offset	TCDn Field	
0x1000 + (32 x n) + 0x00	Source Address (saddr)	
0x1000 + (32 x n) + 0x04	Transfer Attributes (smod, ssize, dmod, dsize)	Signed Source Address Offset (soff)
0x1000 + (32 x n) + 0x08	Signed Minor Loop Offset (smloef, dmloef, mloff)	Inner "Minor" Byte Count (nbytes)
0x1000 + (32 x n) + 0x0c	Last Source Address Adjustment (slast)	
0x1000 + (32 x n) + 0x10	Destination Address (daddr)	
0x1000 + (32 x n) + 0x14	Current "Major" Iteration Count (citer)	Signed Destination Address Offset (doff)
0x1000 + (32 x n) + 0x18	Last Destination Address Adjustment/Scatter Gather Address (dlast_sga)	

Table 210. TCDn 32-bit memory structure (Continued)

0x1000 + (32 x n) + 0x1c	Beginning “Major” Iteration Count (biter)	Channel Control/Status (bwc, major.linkch, done, active, major.e_link, e_sg, d_req, int_half, int_maj, start)
--------------------------	---	--

Figure 200 and Table 211 define word 0 of the TCDn structure, the saddr field.

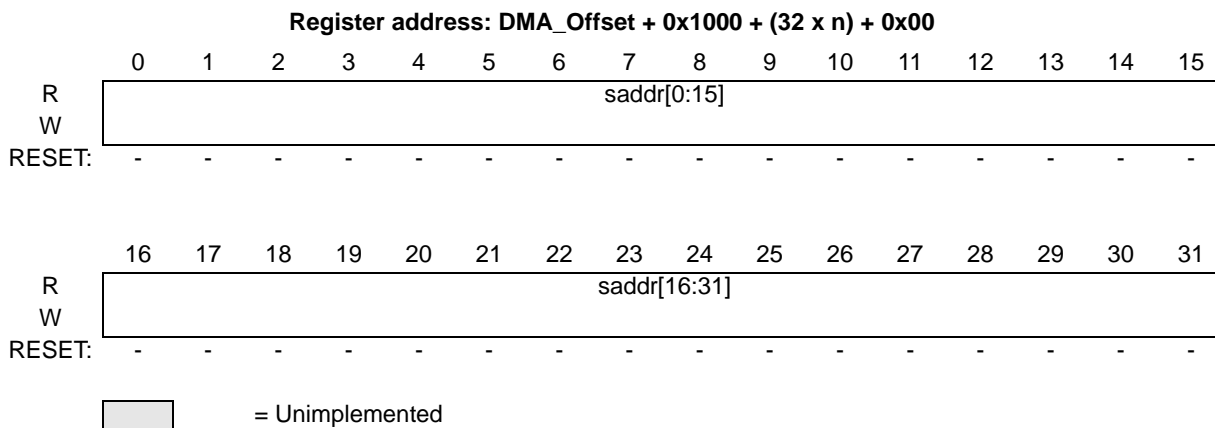


Figure 200. TCDn Word 0 (TCDn.saddr) field

Table 211. TCDn Word 0 (TCDn.saddr) field description

Name	Description	Value
saddr[[0:31]	Source address	Memory address pointing to the source data.

Figure 201 and Table 212 define word 1 of the TCDn structure, the soff and transfer attribute fields.

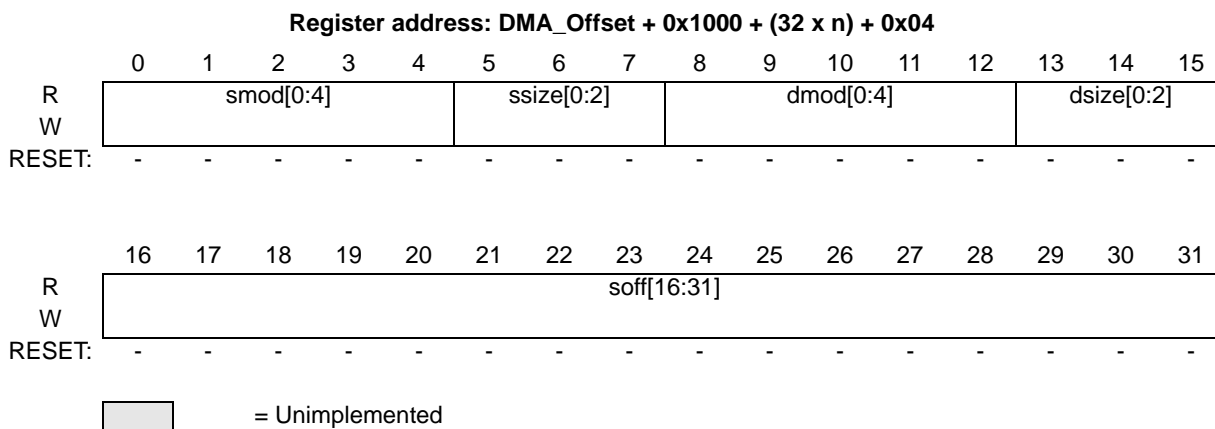


Figure 201. TCDn Word 1 (TCDn.{soff,smod,ssize,dmod,dsize}) fields

Table 212. TCDn Word 1 (TCDn.{smod,ssize,dmod,dsize,soff}) field descriptions

Name	Description	Value
smod[0:4]	Source address modulo	0 Source address modulo feature is disabled. non-0 The value defines a specific address bit which is selected to be either the value after saddr + soff calculation is performed or the original register value. This feature provides the ability to easily implement a circular data queue. For data queues requiring power-of-2 "size" bytes, the queue should be based at a 0-modulo-size address and the smod field set to the appropriate value to freeze the upper address bits. The bit select is defined as $((1 \ll \text{smod}[4:0]) - 1)$ where a resulting 1 in a bit location selects the next state address for the corresponding address bit location and a 0 selects the original register value for the corresponding address bit location. For this application, the soff is typically set to the transfer size to implement post-increment addressing with the smod function constraining the addresses to a 0-modulo-size range.
ssize[0:2]	Source data transfer size	000 8-bit 001 16-bit 010 32-bit 011 Reserved 100 16-byte 101 32-byte 110 Reserved 111 Reserved
dmod[0:4]	Destination address modulo	See the smod[5:0] definition.
dsize[0:2]	Destination data transfer size	See the ssize[2:0] definition.
soff[16:31]	Source address signed offset	Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

Figure 202 and Table 213 define word 2 of the TCDn structure, the nbytes field.

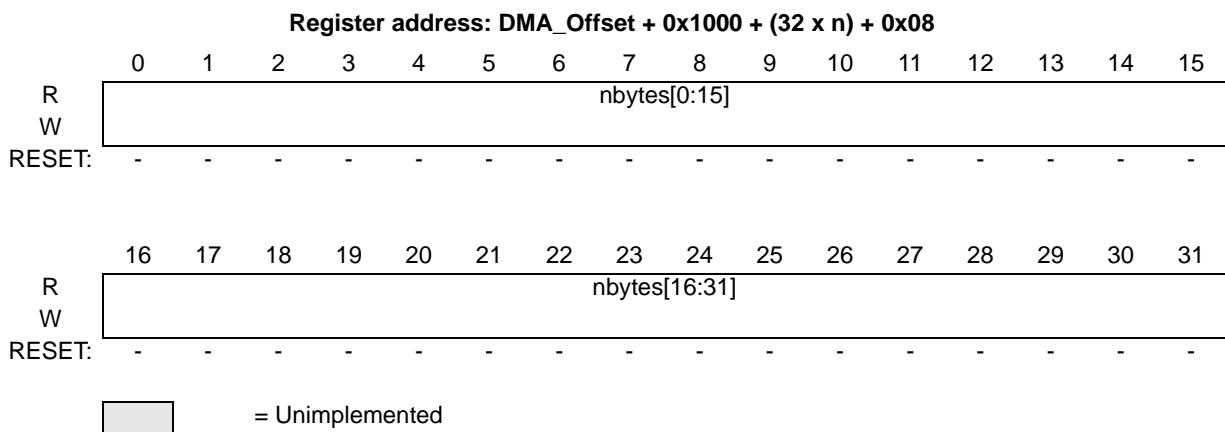


Figure 202. TCDn Word 2 (TCDn.nbytes) field

Table 213. TCDn Word 2 (TCDn.nbytes) field description

Name	Description	Value
nbytes[0:31]	Inner “minor” byte transfer count	<p>Number of bytes to be transferred in each service request of the channel.</p> <p>As a channel is activated, the contents of the appropriate TCD is loaded into the eDMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. After the minor count is exhausted, the current values of the saddr and daddr are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed.</p> <p>The nbytes value 0x0000_0000 is interpreted as 0x1_0000_0000, thus specifying a 4 GB transfer.</p>

When minor loop mapping (DMACR[EMLM] = 1) is enabled, TCD word2 is redefined as four fields: a source minor loop offset enable, a destination minor loop offset enable, a minor loop offset field and a nbytes field.

Figure 203. TCDn Word 2 (TCDn.nbytes) field (DMACR[EMLM] = 1)

Figure 204 and Table 214 define word 3 of the TCDn structure, the slast field.

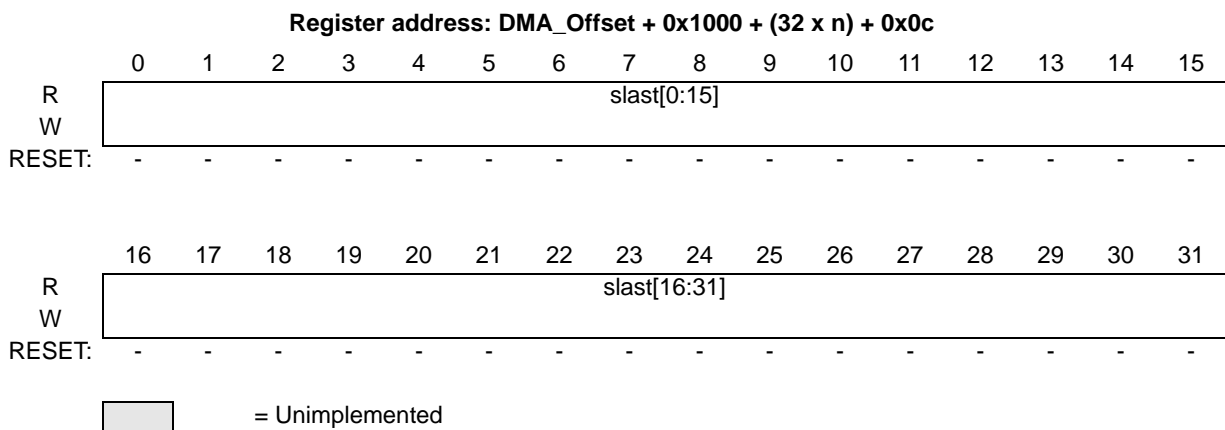


Figure 204. TCDn Word 3 (TCDn.slast) field

Table 214. TCDn Word 3 (TCDn.slast) field descriptions

Name	Description	Value
slast[0:31]	Last source address adjustment	Adjustment value added to the source address at the completion of the outer major iteration count. This value can be applied to “restore” the source address to the initial value, or adjust the address to reference the next data structure.

Figure 205 and Table 215 define word 4 of the TCDn structure, the daddr field.

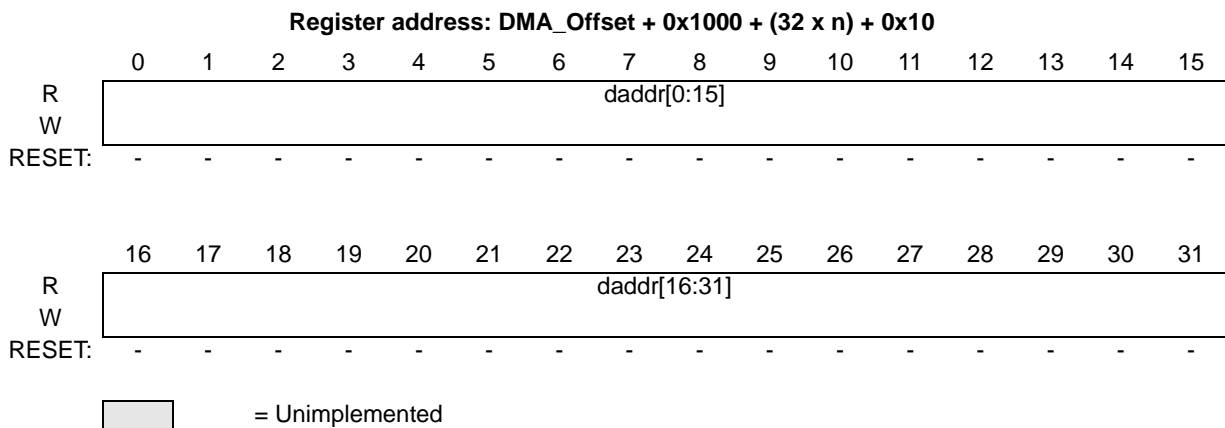


Figure 205. TCDn Word 4 (TCDn.daddr) field

Table 215. TCDn Word 4 (TCDn.daddr) field description

Name	Description	Value
daddr[0:31]	Destination address	Memory address pointing to the destination data.

Figure 206 and Table 216 define word 5 of the TCDn structure, the citer and doff fields.

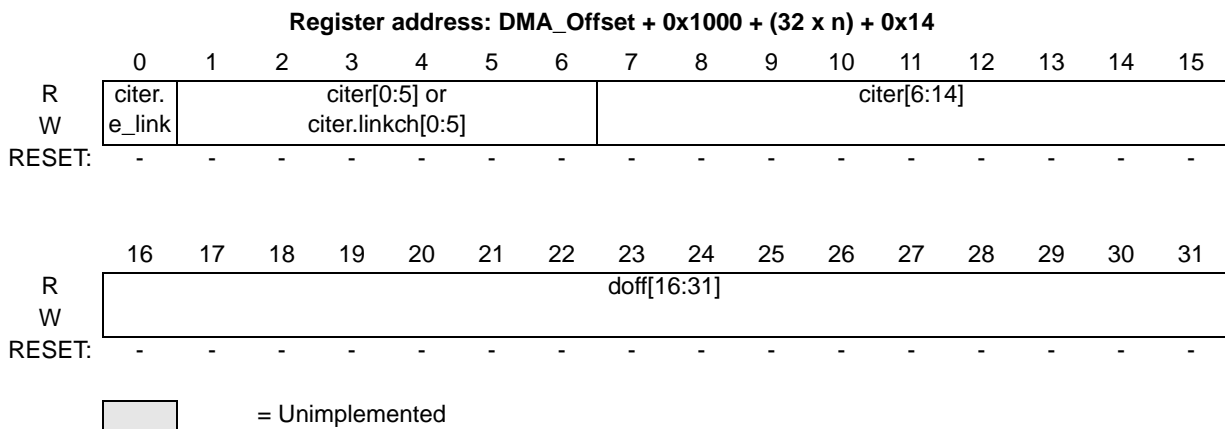


Figure 206. TCDn Word 5 (TCDn.{citer,doff}) fields

Table 216. TCDn Word 5 (TCDn.{doff,citer}) field descriptions

Name	Description	Value
citer.e_link	Enable channel-to-channel linking on minor loop complete	<p>As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by citer.linkch[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.start bit of the specified channel. If channel linking is disabled, the citer value is extended to 15 bits in place of a link channel number. If the "major" loop is exhausted, this link mechanism is suppressed in favor of the major.e_link channel linking. <i>This bit must be equal to the biter.e_link bit otherwise a configuration error will be reported.</i></p> <p>0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.</p>
citer[0:5] or citer.linkch[0:5]	Current "major" iteration count or Link channel number	<p>if (TCD.citer.e_link = 0) then No channel-to-channel linking (or chaining) is performed after the inner "minor" loop is exhausted. TCD word 5, bits [30:25] are used to form a 15 bit citer field.</p> <p>else After the "minor" loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by citer.linkch[5:0] by setting that channel's TCD.start bit.</p> <p>The value contained in citer.linkch[5:0] must not exceed the number of implemented channels.</p>
citer[6:14]	Current "major" iteration count	<p>This 9 or 15-bit count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. Once the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the citer field from the beginning iteration count (biter) field.</p> <p>When the citer field is initially loaded by software, it must be set to the same value as that contained in the biter field.</p> <p>If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001.</p>
doff[16:31]	Destination address signed offset	Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

Figure 207 and Table 217 define word 6 of the TCDn structure, the dlast_sga field.

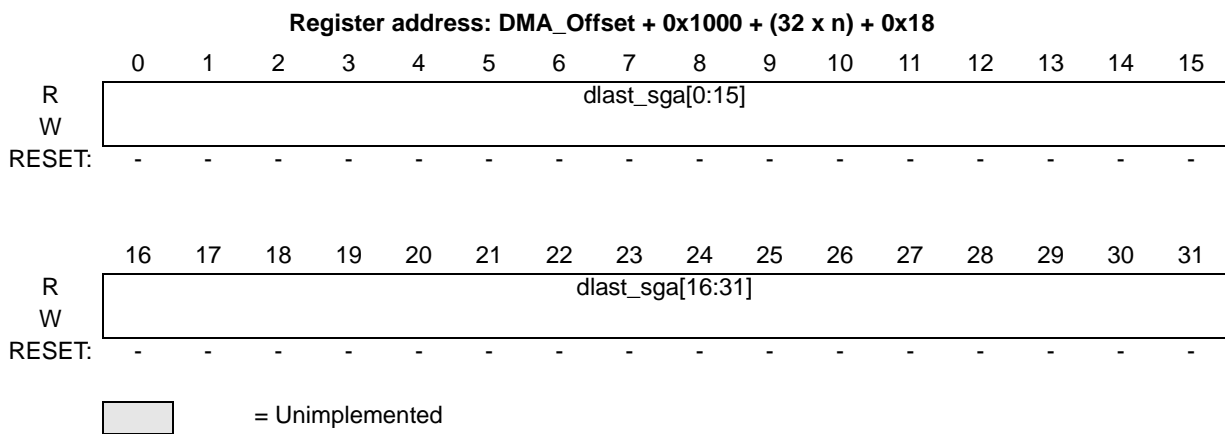


Figure 207. TCDn Word 6 (TCDn.dlast_sga) field

Table 217. TCDn Word 6 (TCDn.dlast_sga) field description

Name	Description	Value
dlast_sga[31:0:0:31]	Last destination address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather)	if (TCD.e_sg = 0) then Adjustment value added to the destination address at the completion of the outer major iteration count. This value can be applied to “restore” the destination address to the initial value, or adjust the address to reference the next data structure. else This address points to the beginning of a 0-modulo-32 region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32, else a configuration error is reported.

Figure 208 and Table 218 define word 7 of the TCDn structure, the biter and control/status fields.

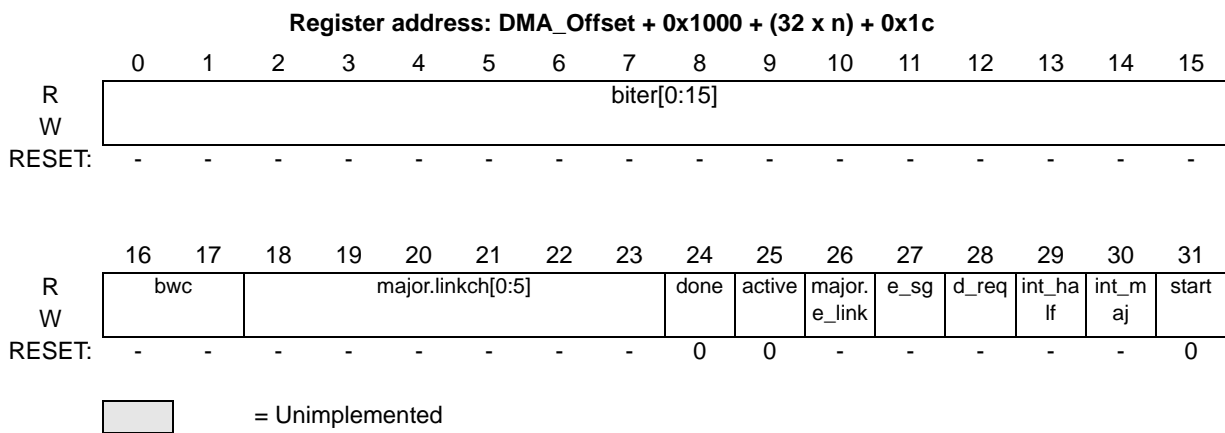


Figure 208. TCDn Word 7 (TCDn.{biter,control/status}) fields

Table 218. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions

Name	Description	Value
biter.e_link	Enable channel-to-channel linking on minor loop complete	<p>This is the initial value copied into the citer.e_link field when the major loop is completed. The citer.e_link field controls channel linking during channel execution. <i>This bit must be equal to the citer.e_link bit otherwise a configuration error will be reported.</i></p> <p>0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.</p>
biter[0:5] or biter.linkch[0:5]	Beginning "major" iteration count or Beginning Link channel number	<p>This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The citer fields controls the iteration count and linking during channel execution.</p> <p>if (TCD.biter.e_link = 0) then No channel-to-channel linking (or chaining) is performed after the inner "minor" loop is exhausted. TCD word 5, bits [30:25] are used to form a 15 bit biter field.</p> <p>else After the "minor" loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by biter.linkch[5:0] by setting that channel's TCD.start bit.</p> <p>The value contained in biter.linkch[5:0] must not exceed the number of implemented channels.</p>
biter[6:14]	Beginning "major" iteration count	<p>This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The citer fields controls the iteration count and linking during channel execution.</p> <p>This 9- or 15-bit count represents the beginning major loop count for the channel. As the major iteration count is exhausted, the contents of the entire 16-bit biter entry is reloaded into the 16-bit citer entry.</p> <p>When the biter field is initially loaded by software, it must be set to the same value as that contained in the citer field.</p> <p>If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001.</p>

Table 218. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (Continued)

Name	Description	Value
bwc[0:1]	Bandwidth control	<p>This two-bit field provides a mechanism to effectively throttle the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the inner minor loop, it continuously generates read/write, read/write, ... sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the platform's cross-bar arbitration switch. To minimize start-up latency, bandwidth control stalls are suppressed for the first two AHB bus cycles and after the last write of each minor loop.</p> <p>The dynamic priority elevation setting elevates the priority of the eDMA as seen by the cross-bar arbitration switch for the executing channel. Dynamic priority elevation is suppressed during the first two AHB bus cycles.</p> <p>00 No eDMA engine stalls 01 Dynamic priority elevation 10 eDMA engine stalls for 4 cycles after each r/w 11 eDMA engine stalls for 8 cycles after each r/w</p>
major.linkch[0:5]	Link channel number	<p>if (TCD.major.e_link = 0) then No channel-to-channel linking (or chaining) is performed after the outer "major" loop counter is exhausted.</p> <p>else After the "major" loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by major.linkch[5:0] by setting that channel's TCD.start bit.</p> <p>The value contained in major.linkch[5:0] must not exceed the number of implemented channels.</p>
done	Channel done	<p>This flag indicates the eDMA has completed the outer major loop. It is set by the eDMA engine as the citer count reaches zero; it is cleared by software, or the hardware when the channel is activated.</p> <p>This bit must be cleared in order to write the major.e_link or e_sg bits.</p>
active	Channel active	<p>This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA engine as the inner minor loop completes or if any error condition is detected.</p>

Table 218. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (Continued)

Name	Description	Value
major.e_link	Enable channel-to-channel linking on major loop complete	<p>As the channel completes the outer major loop, this flag enables the linking to another channel, defined by major.linkch[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.start bit of the specified channel. <i>To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD.done bit is set.</i></p> <p>0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.</p>
e_sg	Enable scatter/gather processing	<p>As the channel completes the outer major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses dlast_sga as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure which is loaded as the transfer control descriptor into the local memory. <i>To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD.done bit is set.</i></p> <p>0 The current channel's TCD is "normal" format. 1 The current channel's TCD specifies a scatter gather format. The dlast_sga field provides a memory pointer to the next TCD to be loaded into this channel after the outer major loop completes its execution.</p>
d_req	Disable request	<p>If this flag is set, the eDMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.</p> <p>0 The channel's DMAERQ bit is not affected. 1 The channel's DMAERQ bit is cleared when the outer major loop is complete.</p>
int_half	Enable an interrupt when major counter is half complete	<p>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is $(citer == (biter \gg 1))$. This halfway point interrupt request is provided to support double-buffered schemes or other types of data movement where the processor needs an early indication of the transfer's progress.</p> <p>0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled. Note: If biter = 1, do not use int_half; use int_major instead.</p>

Table 218. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (Continued)

Name	Description	Value
int_maj	Enable an interrupt when major iteration count completes	<p>If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches zero.</p> <p>0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.</p>
start	Channel start	<p>If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.</p> <p>0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.</p>

19.3 Functional description

This section provides an overview of the microarchitecture and functional operation of the eDMA module.

19.3.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules, which are detailed below.

- eDMA engine
 - addr_path*: This module implements registered versions of two channel transfer control descriptors: channel “x” and channel “y”, and is responsible for all the master bus address calculations. All the implemented channels provide the exact same functionality. This hardware structure allows the data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel service request is asserted while the first channel is active. Once a channel is activated, it runs until the minor loop is completed unless preempted by a higher priority channel. This capability provides a mechanism (optionally enabled by DCHPRIn[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

When any other channel is activated, the contents of its transfer control descriptor is read from the local memory and loaded into the registers of the other *addr_path.channel_{x,y}*. Once the inner minor loop completes execution, the *addr_path* hardware writes the new values for the TCDn.{saddr, daddr, citer} back into the local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCDn.citer field, and a possible fetch of the next TCDn from memory as part of a scatter/gather operation.

- data_path*: This module implements the actual bus master read/write datapath. It includes 32 bytes of register storage (matching the maximum transfer size) and the necessary mux logic to support any required data alignment. The AMBA-AHB read data bus is the primary input, and

the AHB write data bus is the primary output.

The `addr_` and `data_path` modules directly support the 2-stage pipelined AMBA-AHB bus. The `addr_path` module represents the 1st stage of the bus pipeline (the address phase), while the `data_path` module implements the 2nd stage of the pipeline (the data phase).

- *pmodel_charb*: This module implements the first section of eDMA's programming model as well as the channel arbitration logic. The programming model registers are connected to the slave bus (not shown). The `ipd_req[n]` inputs and `dma_ipi_int[n]` outputs are also connected to this module (via the control logic).
- *control*: This module provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read, destination write operations until the number of bytes specified in the inner 'minor loop' byte count has been moved. For descriptors where the sizes are not equal, multiple access of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.
- `transfer_control_descriptor` local memory
 - *memory controller*: This logic implements the required dual-ported controller, handling accesses from both the eDMA engine as well as references from the slave bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the slave bus transaction is stalled. The hooks to a BIST controller for the local TCD memory are included in this module.
 - *memory array*: The TCD is implemented using a single-ported, synchronous compiled RAM memory array

19.3.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in [Figure 209](#), the first segment involves the channel service request. In the diagram, this example uses the assertion of the `ipd_req[n]` signal to request service for channel `n`. Channel service request via software and the `TCDn.start` bit follows the same basic flow as an `ipd_req`. The `ipd_req[n]` input signal is registered internally and then routed to through the eDMA engine, first through the control module, then into the programming model/channel arbitration (`pmodel_charb`) module. In the next cycle, the channel arbitration is performed, either using the fixed-priority or round-robin algorithm. After the arbitration is complete, the activated channel number is sent through the address path (`addr_path`) and converted into the required address to access the TCD local memory. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the `dma_engine.addr_path.channel_{x,y}` registers. The TCD memory is organized 64-bits in width to minimize the time needed to fetch the activated channel's descriptor and load it into the `dma_engine.addr_path.channel_{x,y}` registers.

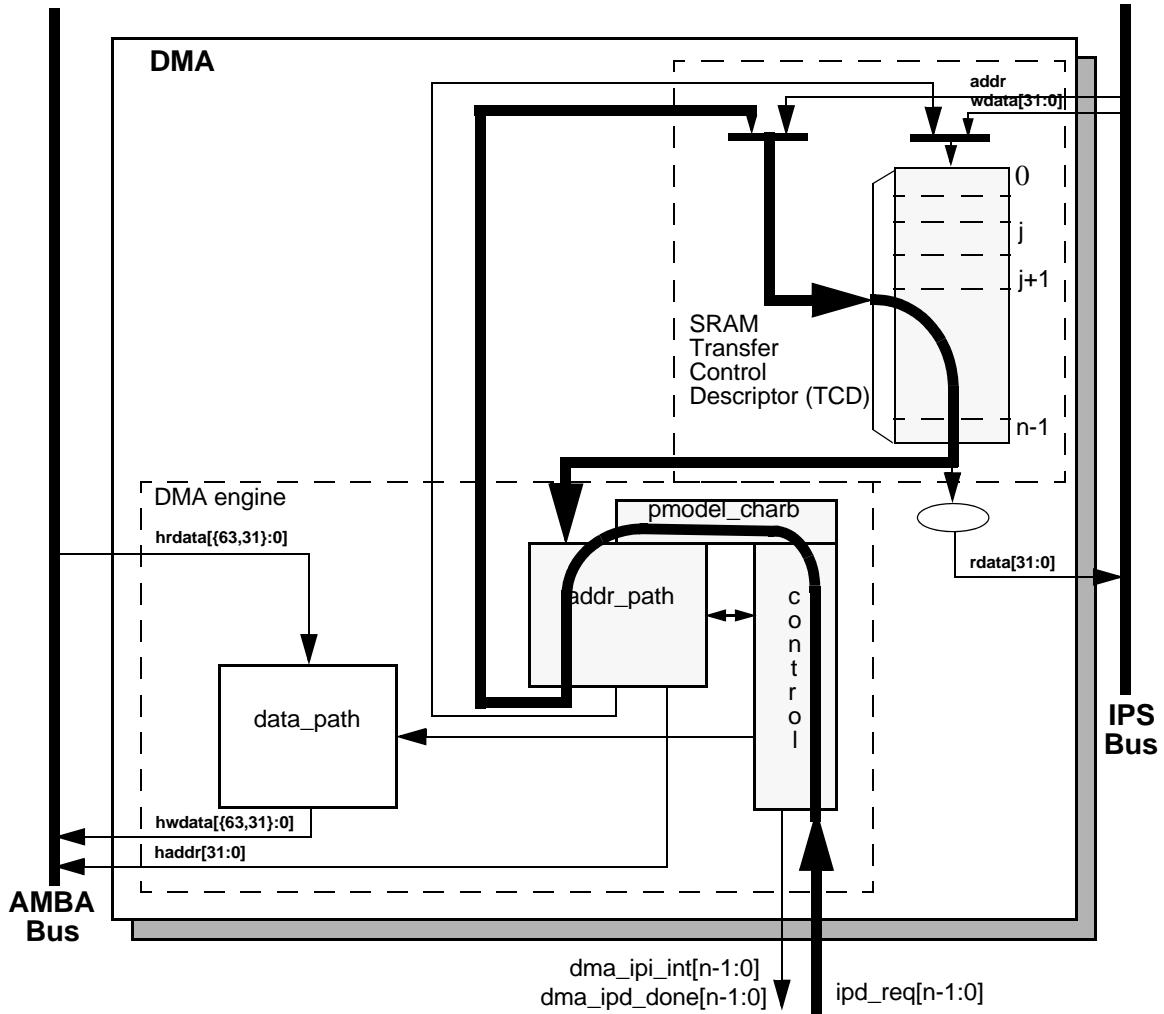


Figure 209. eDMA operation, part 1

In the second part of the basic data flow as shown in [Figure 210](#), the modules associated with the data transfer ($addr_path$, $data_path$ and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the $data_path$ module until it is gated onto the AMBA-AHB bus during the destination write. This source read/destination write processing continues until the inner minor byte count has been transferred. The $dma_ipd_done[n]$ signal is asserted at the end of the minor byte count transfer.

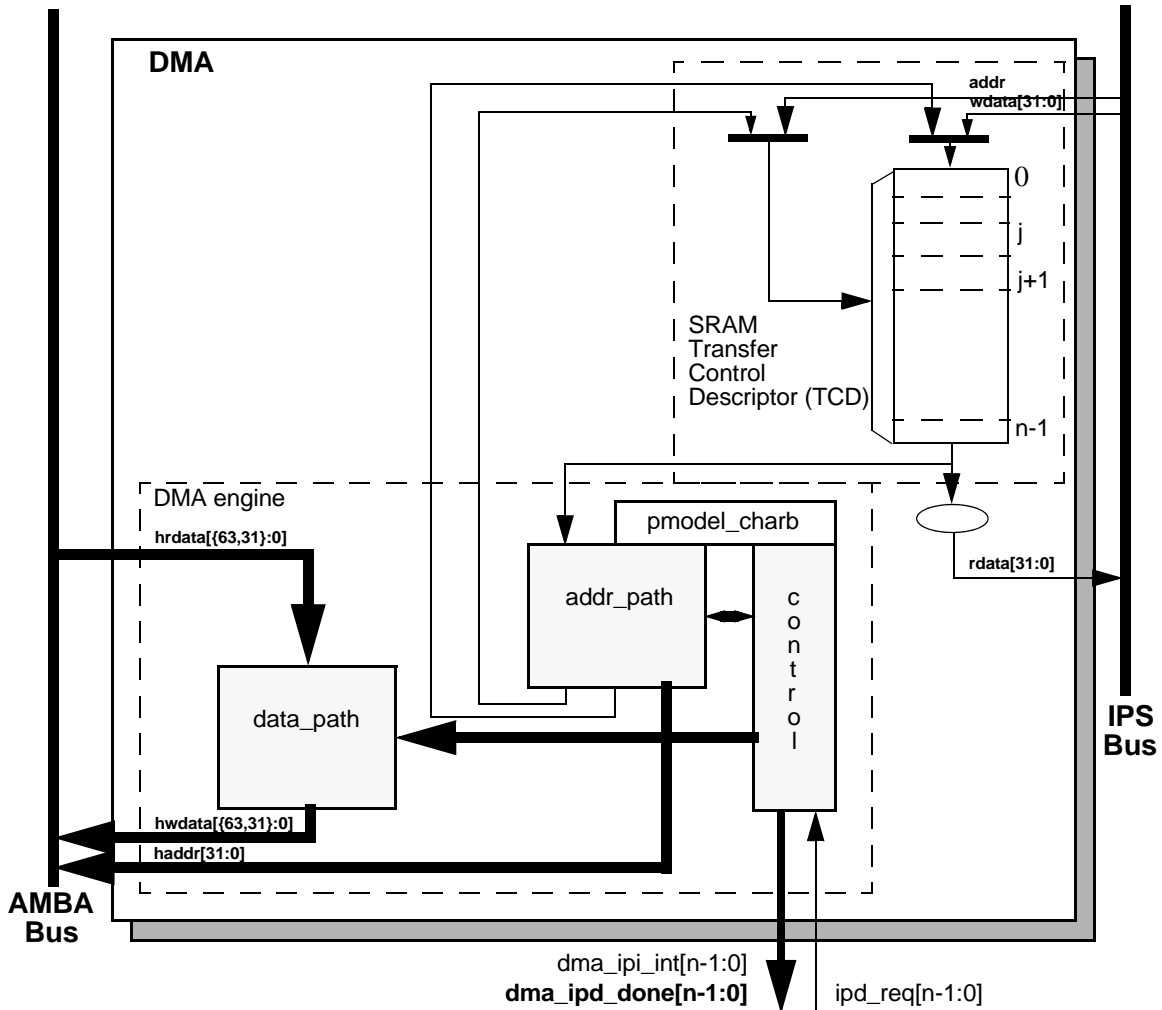


Figure 210. eDMA operation, part 2

Once the inner minor byte count has been moved, the final phase of the basic data flow is performed. In this segment, the `addr_path` logic performs the required updates to certain fields in the channel's TCD, e.g., `saddr`, `daddr`, `citer`. If the outer major iteration count is exhausted, then there are additional operations which are performed. These include the final address adjustments and reloading of the `biter` field into the `citer`. Additionally, assertion of an optional interrupt request occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in [Figure 211](#).

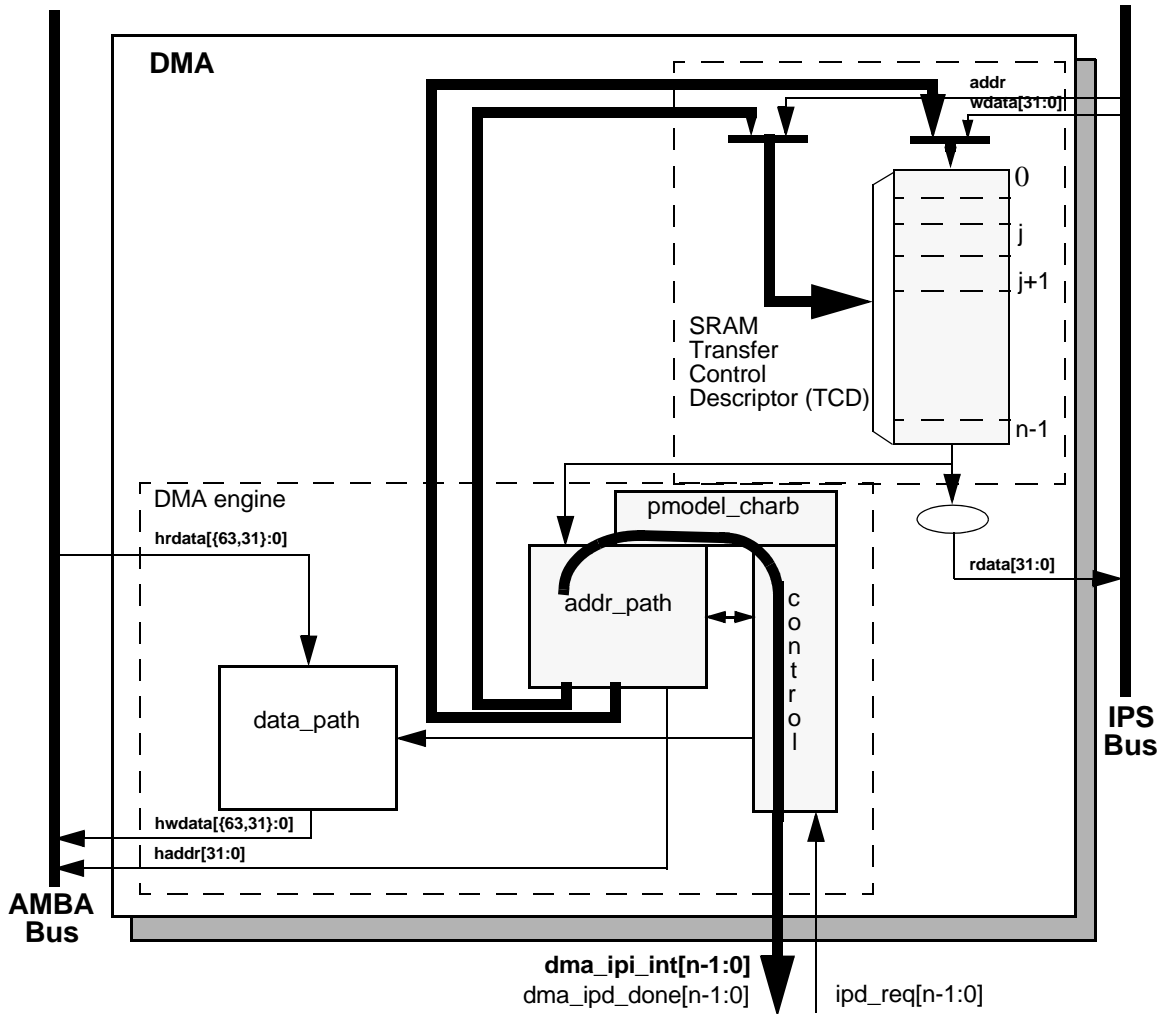


Figure 211. eDMA operation, part 3

19.4 Initialization/application information

19.4.1 eDMA initialization

A typical initialization of the eDMA is:

1. Write the DMACR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRIn registers if a configuration other than the default is desired.
3. Enable error interrupts in the DMAEEI registers if so desired.
4. Write the 32 byte TCD for each channel that may request service.
5. Enable any hardware service requests via the DMAERQ register.
6. Request channel service by either software (setting the TCD.start bit) or by hardware (slave device asserting its ipd_req signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine will read the entire TCD for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the AHB bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD.saddr) to the destination (as defined by the destination address, TCD.daddr) continue until the specified number of bytes (TCD.nbytes) have been transferred. When the transfer is complete, the eDMA engine's local TCD.saddr, TCD.daddr, and TCD.citer are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed, i.e. interrupts, major loop channel linking, and scatter/gather operations, if enabled.

19.4.2 eDMA programming errors

The eDMA performs various tests on the Transfer Control Descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of two errors; Group Priority Error and Channel Priority Error, GPE and CPE in the DMAES register respectively.

For all error types other than Group or Channel Priority Errors, the channel number causing the error is recorded in the DMAES register. If the error source is not removed before the next activation of the problem channel, the error will be detected and recorded again.

The sequence listed below is correct. For item 2, the `dma_ipd_ack{done}` lines will assert only if the selected channel is requesting service via the `ipd_req` signal. I think the typical application will enable error interrupts for all channels. So the user will get an error interrupt, but the channel number for the DMAERR register and the error interrupt request line may be wrong because they reflect the selected channel.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group3 is the highest priority and all channels are unique in that group.
3. Group2 is the next highest priority and has two channels with the same priority level.
4. If Group3 has any service requests, those requests will be executed.
5. Once all of Group3 requests have completed, Group2 will be the next active group.
6. If Group2 has a service request, then an undefined channel in Group2 will be selected and a channel priority error will occur.
7. This will repeat until the all of Group2 requests have been removed or a higher priority Group3 request comes in.

A group priority error is global and any request in any group will cause a group priority error.

In general, if priority levels are not unique, the highest (channel/group) priority that has an active request will be selected, but the lowest numbered (channel/group) with that priority will be selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts and error reporting will be associated with the selected channel.

19.4.3 eDMA arbitration mode considerations

19.4.3.1 Fixed group arbitration, fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group will be selected to execute. If the eDMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller — i.e. no other groups will be serviced if there is always at least one eDMA request pending on a channel in the highest priority group when the controller arbitrates the next eDMA request.

The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

19.4.3.2 Round-robin group arbitration, fixed channel arbitration

The occurrence of one or more eDMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with an service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round robin algorithm will select the highest pending request from the next group in the round robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced.

The advantage of this scenario is that no one group uses all the eDMA bandwidth.

The highest priority channel selection latency is potentially greater than fixed/fixed arbitration.

Excessive request rates on high priority channels can prevent the servicing of lower priority channels in the same group.

19.4.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in [Section 19.4.3.2, Round-robin group arbitration, fixed channel arbitration](#), but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round robin manner, any channel that generates eDMA requests faster than a combination of the group round robin service rate and the channel service rate for its group will not

prevent the servicing of other channels in its group. Any eDMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high.

All channels are treated equally. Priority levels are not used in round robin/round robin mode.

19.4.3.4 Fixed group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in [Section 19.4.3.1, Fixed group arbitration, fixed channel arbitration](#), but all the channels in the highest priority group will be serviced.

Service latency will be short on the highest priority group, but can become much longer as the group priority decreases.

19.4.4 eDMA transfer

19.4.4.1 Single request

To perform a simply transfer of ‘n’ bytes of data with one activation, set the major loop to one (TCD.citer = TCD.biter = 1). The data transfer will begin after the channel service request is acknowledged and the channel is selected to execute. Once the transfer is complete, the TCD.done bit will be set and an interrupt will be generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCD.citer    = TCD.biter = 1
TCD.nbytes  = 16
TCD.saddr   = 0x1000
TCD.soff    = 1
TCD.ssize   = 0
TCD.sldest  = -16
```

```

TCD.daddr    = 0x2000
TCD.doff     = 4
TCD.dsize    = 2
TCD.dlast_sga= -16
TCD.int_maj  = 1
TCD.start    = 1 (TCD.word7 should be written last after all other fields have been initialized)
All other TCD fields = 0

```

This generates the following sequence of events:

1. slave bus write to the TCD.start bit requests channel service
2. The channel is selected by arbitration for servicing
3. eDMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
 - a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
 - b. write_word(0x2000) -> *first iteration of the minor loop*
 - c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
 - d. write_word(0x2004) -> *second iteration of the minor loop*
 - e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
 - f. write_word(0x2008) -> *third iteration of the minor loop*
 - g. read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
 - h. write_word(0x200c) -> *last iteration of the minor loop -> major loop complete*
6. eDMA engine writes: TCD.saddr = 0x1000, TCD.daddr = 0x2000, TCD.citer = 1 (TCD.biter)
7. eDMA engine writes: TCD.active = 0, TCD.done = 1, DMAINT[n] = 1
8. The channel retires

The eDMA goes idle or services next channel.

19.4.4.2 Multiple requests

The next example is the same as the previous example, with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The DMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests is enabled in the DMAERQ register, channel service requests are initiated by the slave device.

```
TCD.citer    = TCD.biter = 2
```

TCD.slstart = -32

TCD.dlast_sga = -32

This would generate the following sequence of events:

1. First hardware (ipd_req) request for channel service
2. The channel is selected by arbitration for servicing
3. eDMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
 - a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
 - b. write_word(0x2000) -> *first iteration of the minor loop*
 - c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
 - d. write_word(0x2004) -> *second iteration of the minor loop*
 - e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
 - f. write_word(0x2008) -> *third iteration of the minor loop*
 - g. read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
 - h. write_word(0x200c) -> *last iteration of the minor loop*
6. eDMA engine writes: TCD.saddr = 0x1010, TCD.daddr = 0x2010, TCD.citer = 1
7. eDMA engine writes: TCD.active = 0
8. The channel retires -> *one iteration of the major loop*

The eDMA goes idle or services next channel.

9. Second hardware (ipd_req) requests channel service
10. The channel is selected by arbitration for servicing
11. eDMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
12. eDMA engine reads: channel TCD data from local memory to internal register file
13. The source to destination transfers are executed as follows:
 - a. read_byte(0x1010), read_byte(0x1011), read_byte(0x1012), read_byte(0x1013)
 - b. write_word(0x2010) -> *first iteration of the minor loop*
 - c. read_byte(0x1014), read_byte(0x1015), read_byte(0x1016), read_byte(0x1017)
 - d. write_word(0x2014) -> *second iteration of the minor loop*
 - e. read_byte(0x1018), read_byte(0x1019), read_byte(0x101a), read_byte(0x101b)
 - f. write_word(0x2018) -> *third iteration of the minor loop*
 - g. read_byte(0x101c), read_byte(0x101d), read_byte(0x101e), read_byte(0x101f)
 - h. write_word(0x201c) -> *last iteration of the minor loop -> major loop complete*

14. eDMA engine writes: TCD.saddr = 0x1000, TCD.daddr = 0x2000, TCD.citer = 2 (TCD.biter)
15. eDMA engine writes: TCD.active = 0, TCD.done = 1, DMAINT[n] = 1
16. The channel retires -> *major loop complete*

The eDMA goes idle or services the next channel.

19.4.5 TCD status

19.4.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD.citer field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD.start bit AND the TCD.active bit. The minor loop complete condition is indicated by both bits reading zero after the TCD.start was written to a one. Polling the TCD.active bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

1. TCD.start = 1, TCD.active = 0, TCD.done = 0 (channel service request via software)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle)
or
TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD.citer field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware activated channel:

1. ipd_req asserts (channel service request via hardware)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle)
or
TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

For both activation types, the major loop complete status is explicitly indicated via the TCD.done bit.

The TCD.start bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

19.4.5.2 Active channel TCD reads

The eDMA will read back the 'true' TCD.saddr, TCD.daddr, and TCD.nbytes values if read while a channel is executing. The 'true' values of the saddr, daddr, and nbytes are the values the eDMA engine is currently using in its internal register file and not the values in the TCD local memory for that channel.

The addresses (saddr and daddr) and nbytes (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

19.4.5.3 Preemption status

Preemption is only available when *fixed* arbitration is selected for *both* group and channel arbitration modes. A preempt-able situation is one in which a preempt-enabled channel is running and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when round-robin arbitration mode is selected.

The TCD.active bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. Two TCD.active bits set at the same time in the overall TCD map indicates a higher priority channel is actively preempting a lower priority channel.

The worst case latency when switching to a preempt channel is the summation of:

- arbitration latency (2 cycles)
- bandwidth control stalls (if enabled)
- the time to execute two read/write sequences (including AHB bus holds; a system dependency driven by the slave devices or the crossbar)

19.4.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD.start bit of another channel (or itself) thus initiating a service request for that channel. This operation is automatically performed by the eDMA engine at the conclusion of the major or minor loop when properly enabled.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD.citer.e_link field are used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, with the initial fields of:

TCD.citer.e_link = 1

TCD.citer.linkch = 0xC

TCD.citer value = 0x4

TCD.major.e_link = 1

TCD.major.linkch = 0x7

will execute as:

1. minor loop done -> set channel 12 TCD.start bit
2. minor loop done -> set channel 12 TCD.start bit



3. minor loop done -> set channel 12 TCD.start bit
4. minor loop done, major loop done -> set channel 7 TCD.start bit

When minor loop linking is enabled (TCD.citer.e_link = 1), the TCD.citer field uses a nine bit vector to form the current iteration count.

When minor loop linking is disabled (TCD.citer.e_link = 0), the TCD.citer field uses a 15 bit vector to form the current iteration count. The bits associated with the TCD.citer.linkch field are concatenated onto the citer value to increase the range of the citer.

NOTE

The TCD.citer.e_link bit and the TCD.biter.e_link bit must equal or a configuration error will be reported. The citer and biter vector widths must be equal to calculate the major loop, half-way done interrupt point.

19.4.7 Dynamic programming

19.4.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to round-robin channel arbitration mode, change the channel priorities, then switch back to fixed arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to round-robin group arbitration mode, change the group priorities, then switch back to fixed arbitration mode.
2. Disable all channels, change the group priorities, then enable the appropriate channels.

19.4.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in [Table 219](#) is recommended when executing a dynamic channel link request.

Table 219. Coherency model for a dynamic channel link request

Step	Action
1	Write 1b to the TCD.major.e_link bit.
2	Read back the TCD.major.e_link bit.
3	Test the TCD.major.e_link request status: <ul style="list-style-type: none"> • If TCD.major.e_link = 1b, the dynamic link attempt was successful. • If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

19.4.7.3 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

19.4.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in [Table 220](#) may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Table 220. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to the TCD.d_req bit. Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
3	Write the TCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.
5	Read back the 16 bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: <ul style="list-style-type: none"> • If e_sg = 1b, the dynamic link attempt was successful. • If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). • If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

19.4.7.3.2 Method 2 (channel using major loop linking)

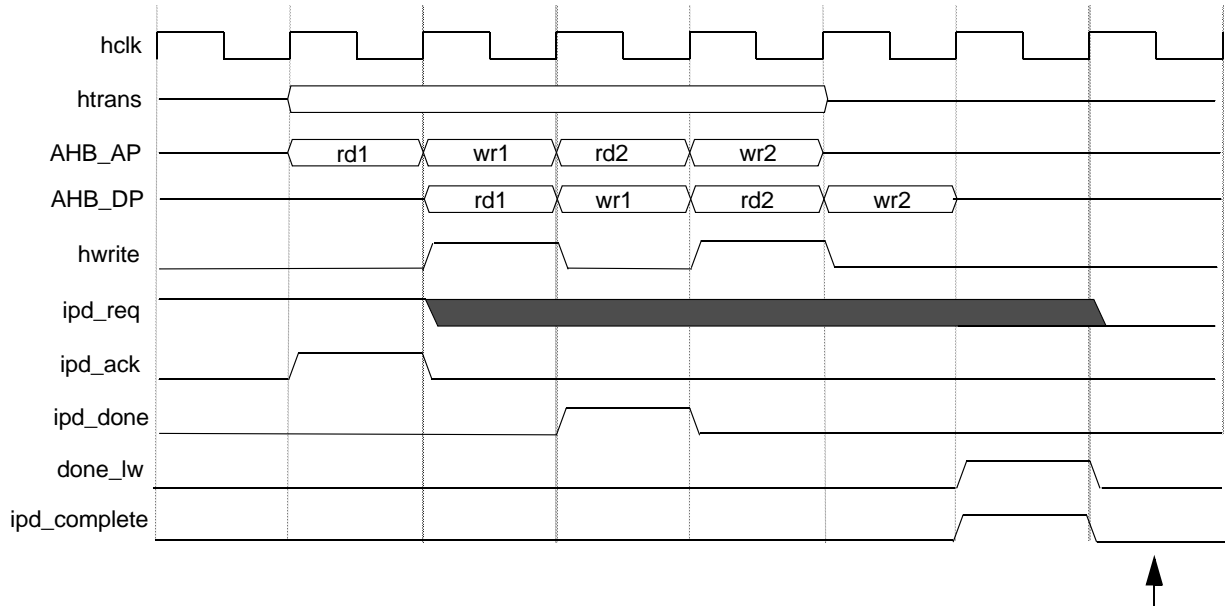
For a channel using major loop channel linking, the coherency model in [Table 221](#) may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 221. Coherency model for method 2

Step	Action
1	Write 1b to the TCD.d_req bit. Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
2	Write the TCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: <ul style="list-style-type: none"> • If e_sg = 1b, the dynamic link attempt was successful. • If e_sg = 0b, read the 32 bit TCD dlast_sga field. • If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). • If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

19.4.8 Hardware request release timing

This section provides a timing diagram for deasserting the ipd_req hardware request signal. Figure 212 shows two read write sequences with grey indicating the release of the ipd_req hardware request signal.



Note: ipd_req must de-assert in this cycle unless another service request is intended

Figure 212. ipd_req hardware handshake

Chapter 20 DMACHMUX

20.1 Introduction

20.1.1 Overview

The DMA Mux allows to route 21 DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in [Figure 213](#).

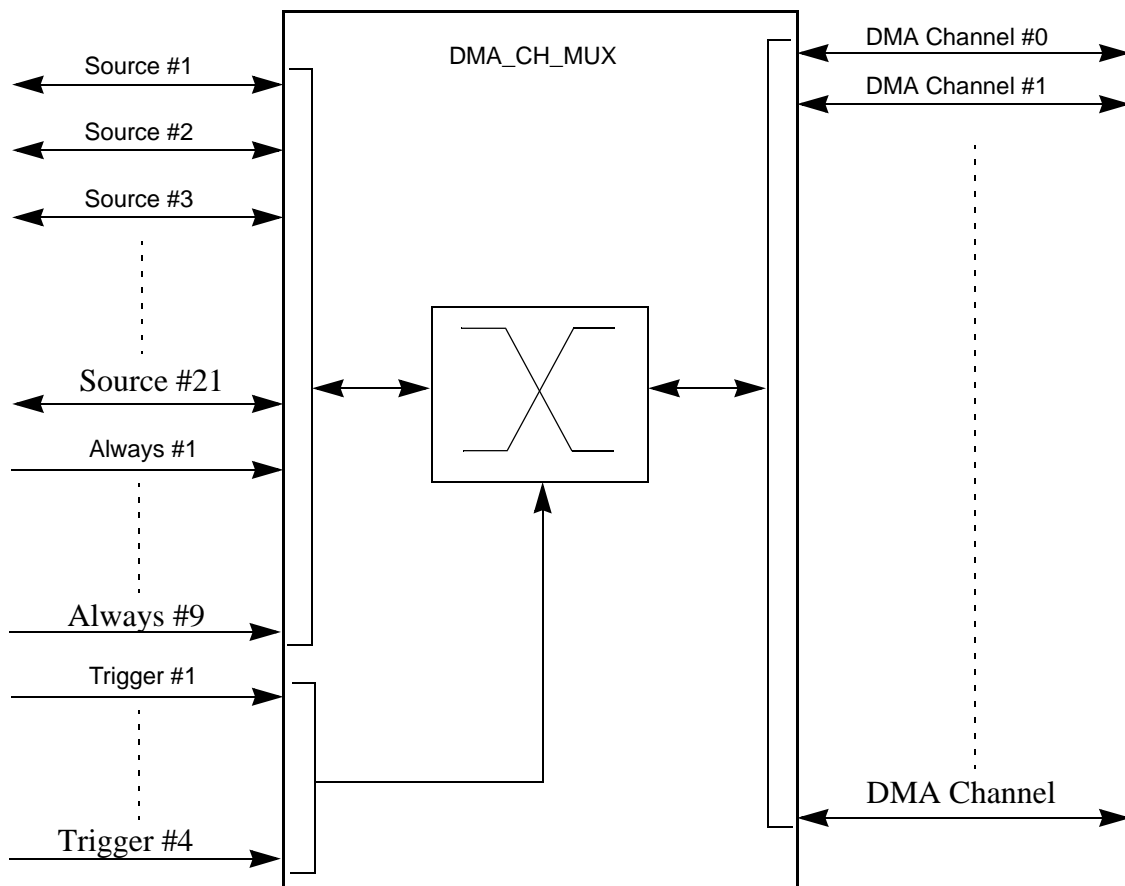


Figure 213. DMA Mux Block Diagram

20.1.2 Features

The DMA Channel Mux provides these features:

- 21 peripheral slots + 9 always-on slots can be routed to 16 channels
- 16 independently selectable DMA channels routers
 - the first 4 channels additionally provide a trigger functionality
- Each channel router can be assigned to one of 21 possible peripheral DMA slots or to one of the 9 always-on slots.

20.1.3 Modes of Operation

The following operation modes are available:

- Disabled Mode

In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA Channel Mux. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (e.g. changing the period of a DMA trigger).
- Normal Mode

In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMA Mux in this mode is completely transparent to the system.
- Periodic Trigger Mode

In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0-4.

20.2 External Signal Description

20.2.1 Overview

The DMA Mux has no external pins.

20.3 Memory Map and Register Definition

This section provides a detailed description of all memory-mapped registers in the DMA Mux.

[Table 222](#) shows the memory map for the DMA Mux. Note that all addresses are offsets; the absolute address may be computed by adding the specified offset to the base address of the DMA Mux.

Table 222. DMA_MUX memory map

Offset from DMA_MUX_BASE (0xFFFFD_C000)	Register	Access	Reset value	Location
0x00	Channel #0 Configuration (CHCONFIG0)	R/W	0x00	on page 437
0x01	Channel #1 Configuration (CHCONFIG1)	R/W	0x00	on page 437

Table 222. DMA_MUX memory map (continued)

Offset from DMA_MUX_BASE (0xFFFD_C000)	Register	Access	Reset value	Location
0x02	Channel #2 Configuration (CHCONFIG2)	R/W	0x00	on page 437
0x03	Channel #3 Configuration (CHCONFIG3)	R/W	0x00	on page 437
0x04	Channel #4 Configuration (CHCONFIG4)	R/W	0x00	on page 437
0x05	Channel #5 Configuration (CHCONFIG5)	R/W	0x00	on page 437
0x06	Channel #6 Configuration (CHCONFIG6)	R/W	0x00	on page 437
0x07	Channel #7 Configuration (CHCONFIG7)	R/W	0x00	on page 437
0x08	Channel #8 Configuration (CHCONFIG8)	R/W	0x00	on page 437
0x09	Channel #9 Configuration (CHCONFIG9)	R/W	0x00	on page 437
0x0A	Channel #10 Configuration (CHCONFIG10)	R/W	0x00	on page 437
0x0B	Channel #11 Configuration (CHCONFIG11)	R/W	0x00	on page 437
0x0C	Channel #12 Configuration (CHCONFIG12)	R/W	0x00	on page 437
0x0D	Channel #13 Configuration (CHCONFIG13)	R/W	0x00	on page 437
0x0E	Channel #14 Configuration (CHCONFIG14)	R/W	0x00	on page 437
0x0F	Channel #15 Configuration (CHCONFIG15)	R/W	0x00	on page 437
0x001F–0x3FFF	Reserved			

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCONFIG0 through CHCONFIG3 are accessible by a 32-bit READ/WRITE to address ‘Base + 0x00’, but performing a 32-bit access to address ‘Base + 0x01’ is illegal.

20.3.1 Register Descriptions

The following memory-mapped registers are available in the DMA Channel Mux.

20.3.1.1 Channel Configuration Registers

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

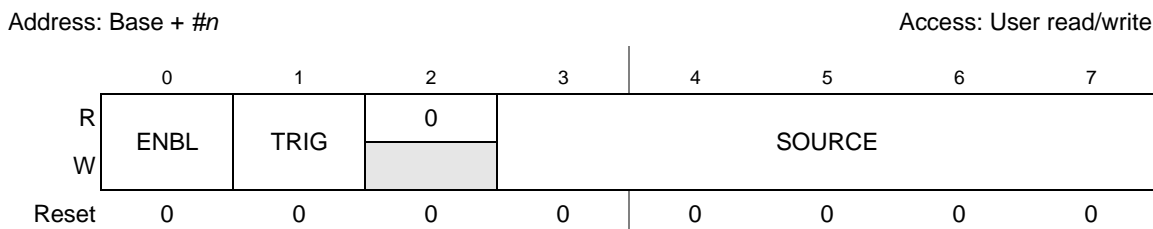


Figure 214. Channel Configuration Registers (CHCONFIG#n)

Table 223. CHCONFIGxx Field Descriptions

Field	Description
0 ENBL	DMA Channel Enable. ENBL enables the DMA Channel 0 DMA channel is disabled. This mode is primarily used during configuration of the DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or re-configure a DMA channel. 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable (for triggered channels only). TRIG enables the periodic trigger capability for the DMA Channel 0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel. 1 Triggering is enabled
3–7 SOURCE	DMA Channel Source (slot). SOURCE specifies which DMA source, if any, is routed to a particular DMA channel.

Table 224. Channel and Trigger Enabling

ENBL	TRIG	Function	Mode
0	X	DMA Channel is disabled	Disabled Mode
1	0	DMA Channel is enabled with no triggering (transparent)	Normal Mode
1	1	DMA Channel is enabled with triggering	Periodic Trigger Mode

NOTE

Setting multiple CHCONFIG registers with the same Source value will result in unpredictable behavior.

NOTE

Before changing the trigger or source settings a DMA channel must be disabled via the CHCONFIG[#n].ENBL bit.

20.4 DMA request mapping

This sections defines the integration of the DMA channel multiplexer. [Table 225](#), shows which modules are connected to which DMA multiplexer slot. [Table 226](#) shows the trigger inputs.

Table 225. DMA channel mapping

DMA Channel	Resource	Module	DMA Requesting Module	DMA Mux Input
1	DSPI_TFFF	DSPI 0	DSPI_0 TX	DMA MUX Source #1
2	DSPI_RFDF	DSPI 0	DSPI_0 RX	DMA MUX Source #2
3	DSPI_TFFF	DSPI 1	DSPI_1 TX	DMA MUX Source #3
4	DSPI_RFDF	DSPI 1	DSPI_1 RX	DMA MUX Source #4
5	DSPI_TFFF	DSPI 2	DSPI_2 TX	DMA MUX Source #5
6	DSPI_RFDF	DSPI 2	DSPI_2 RX	DMA MUX Source #6
8	SAI_TFIFO	SAI_0	SAI_TX	DMA MUX Source #8
9	SAI_RFIFO	SAI_0	SAI_RX	DMA MUX Source #9
10	SAI_TFIFO	SAI_1	SAI_TX	DMA MUX Source #10
11	SAI_RFIFO	SAI_1	SAI_RX	DMA MUX Source #11
12	SAI_TFIFO	SAI_2	SAI_TX	DMA MUX Source #12
13	SAI_RFIFO	SAI_2	SAI_RX	DMA MUX Source #13
16	channel0	etimer0	eTimer_0 CH1	DMA MUX Source #16
17	channel1	etimer0	eTimer_0 CH2	DMA MUX Source #17
20	DMA	adc0	ADC_0	DMA MUX Source #20
21	Always requestor	—	—	—
22	Always requestor	—	—	—
23	Always requestor	—	—	—
24	Always requestor	—	—	—
25	Always requestor	—	—	—
26	Always requestor	—	—	—
27	Always requestor	—	—	—
28	Always requestor	—	—	—
29	Always requestor	—	—	—
30	Always requestor	—	—	—

Table 226. DMA Trigger Mapping

DMACHMUX Channel Trigger	Source Module	Source Signal
0	PIT	Trigger Channel 0
1	PIT	Trigger Channel 1
2	PIT	Trigger Channel 2
3	PIT	Trigger Channel 3

The triggers can be used to gate the request signals from the individual IP modules. Using the periodic timer impulses limits the peak bandwidth of an individual DMA channel, to prevent delay of other transactions.

20.5 Functional Description

This section provides a functional description of the DMA Mux. The primary purpose of the DMA Mux is to provide flexibility in the system’s use of the available DMA channels. As such, configuration of the DMA Mux is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in [Section 20.6.2, “Enabling and Configuring Sources”](#) is followed, the configuration of the DMA MUX may be changed during the normal operation of the system.

Functionally, the DMA Mux channels may be divided into two classes: Channels, which implement the normal routing functionality plus periodic triggering capability, and channels, which implement only the normal routing functionality.

20.5.1 DMA Channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMA Mux provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. Please refer to the Periodic Interrupt Timer Block Guide for more information on this topic.

NOTE

Because of the dynamic nature of the system (i.e. DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

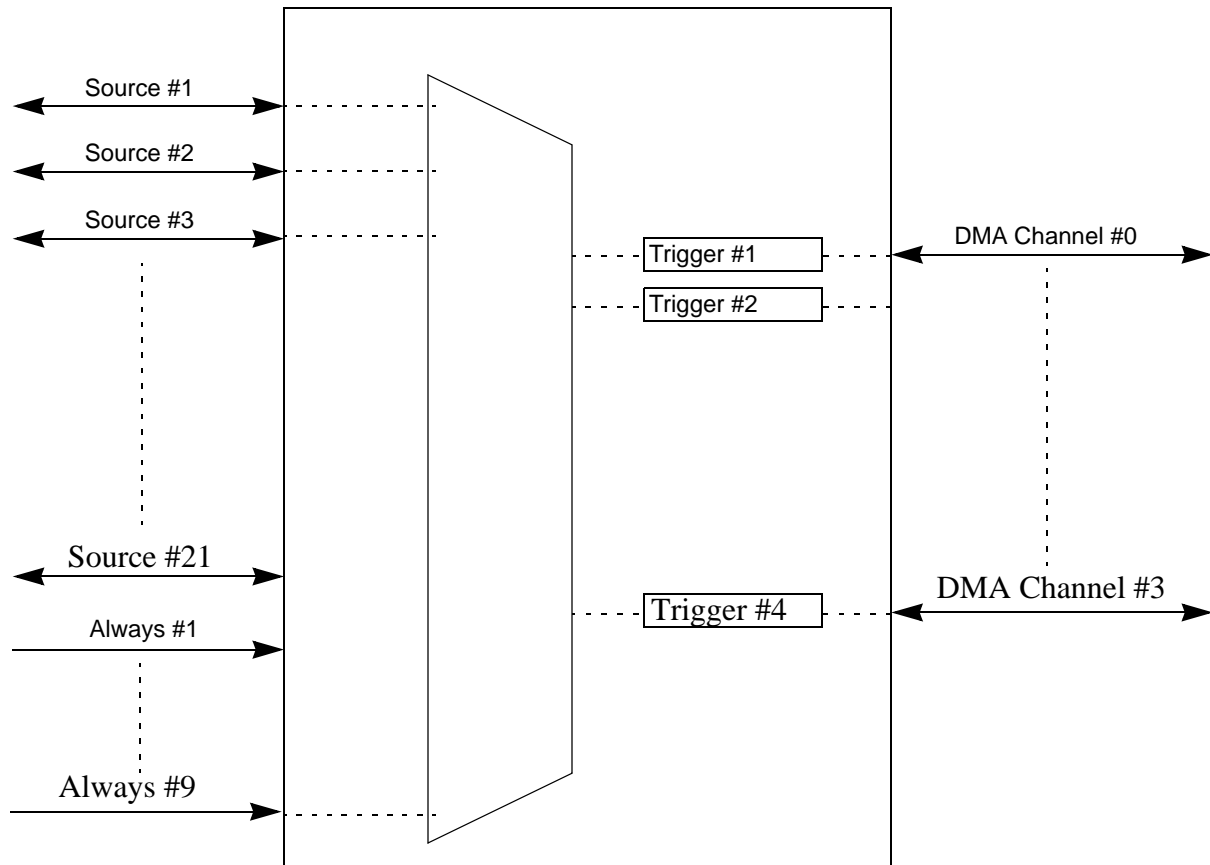


Figure 215. DMA Mux triggered channels

The DMA channel triggering capability allows the system to “schedule” regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in [Figure 216](#).

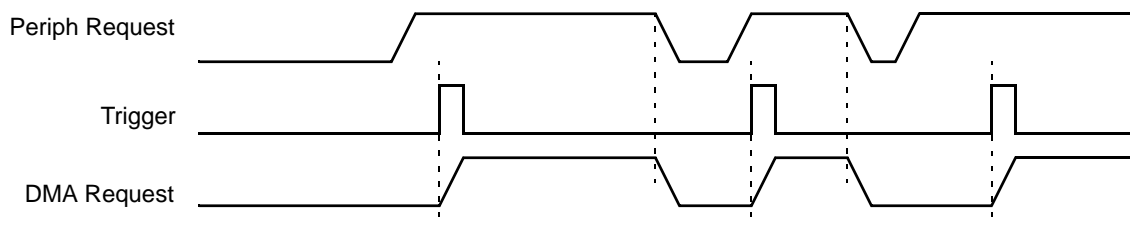


Figure 216. DMA Mux Channel Triggering: Normal Operation

Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in [Figure 217](#).

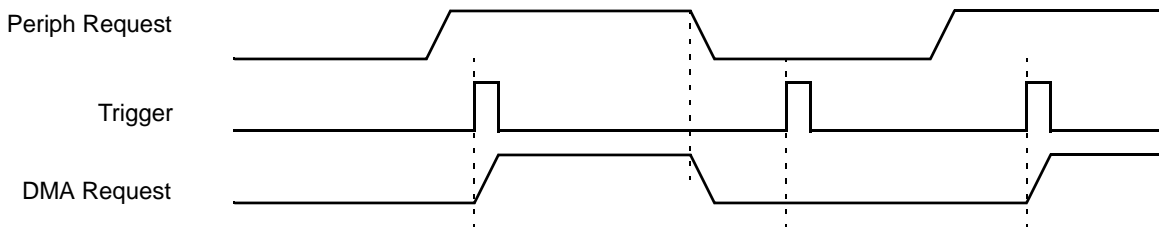


Figure 217. DMA Mux Channel Triggering: Ignored Trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 μ s (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (i.e.-resolution, range of values, etc.) may be found in the Periodic Interrupt Timer (PIT) Block Guide.

20.5.2 DMA Channels with no triggering capability

The other channels of the DMA Mux provide the normal routing functionality as described in [Section 20.1.3, “Modes of Operation”](#).

20.5.3 "Always Enabled" DMA Sources

In addition to the peripherals that can be used as DMA sources, there are 9 additional DMA sources that are "always enabled". Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the "always enabled" sources provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO - Moving data from/to one or more GPIO pins, either un-throttled (i.e.-as fast as possible), or periodically (using the DMA triggering capability).
- Doing DMA transfers from memory to memory - Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice-versa) - Similar to memory to memory transfers, this is typically done as quickly as possible.

- Any DMA transfer that requires software activation - Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, a "always enabled" DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require a new "start" event be sent. This can either be a new software activation, or a transfer request from the DMA Channel Mux. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the DMA to transfer all of the data in a single minor loop (i.e.-major loop counter = 1), no re-activation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMA Channel Mux.
- Use explicit software re-activation. In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to re-activate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMA Channel Mux.
- Use a "always enabled" DMA source. In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA Channel Mux does the channel re-activation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the re-activation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

20.6 Initialization/Application Information

20.6.1 Reset

The reset state of each individual bit is shown within the Register Description section (See [Section 20.3.1, "Register Descriptions"](#)). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

20.6.2 Enabling and Configuring Sources

Enabling a source with periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point
4. Configure the corresponding timer
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set

Example 1. Configure source #5 Transmit for use with DMA Channel 2, with periodic triggering capability

1. Write 0x00 to CHCONFIG2 (Base Address + 0x02)
2. Configure Channel 2 in the DMA, including enabling the channel
3. Configure a timer for the desired trigger interval
4. Write 0xC5 to CHCONFIG2 (Base Address + 0x02)

The following code example illustrates steps #1 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR      0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0xC5;
```

Enabling a source without periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL is set and the TRIG bit is cleared

Example 2. Configure source #5 Transmit for use with DMA Channel 2, with no periodic triggering capability.

1. Write 0x00 to CHCONFIG2 (Base Address + 0x02)
2. Configure Channel 2 in the DMA, including enabling the channel
3. Write 0x85 to CHCONFIG2 (Base Address + 0x02)

The following code example illustrates steps #1 and #3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR      0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0x85;
```

Disabling a source

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCONFIG registers. Additionally, some module specific configuration may be necessary. Please refer to the appropriate section for more details.

Switching the source of a DMA Channel

1. Disable the DMA channel in the DMA and re-configure the channel for the new source
2. Clear the ENBL and TRIG bits of the DMA channel
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set

Example 3. Switch DMA Channel 8 from source #5 transmit to source #7 transmit

1. In the DMA configuration registers, disable DMA channel 8 and re-configure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCONFIG8 (Base Address + 0x08)
3. Write 0x87 to CHCONFIG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channels 8 does not support the periodic triggering functionality).

The following code example illustrates steps #2 and #3 above:

```
In File registers.h:
```

```

#define DMAMUX_BASE_ADDR      0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
    :
    :
*CHCONFIG8 = 0x00;
*CHCONFIG8 = 0x87;

```

20.6.3 Freezing in STOP and HALT mode

If a DMA capable peripheral is programmed to run on divided system clock, then do not configure DMACHMUX to be frozen in STOP/HALT mode using the DMACHMUX PCTL register.

Chapter 21

Video Encoder Wrapper

21.1 Introduction

Figure 218 shows the block diagram of video encoder.

The video encoder accepts either an ITU-BT656 like compatible video stream with embedded sync signals, or a video stream with external *vsync* and *href* signals on its parallel interface. The encoder downsamples the stream to 4:2:0 format, compresses it using JPEG encoding, and then stores it in the output buffer.

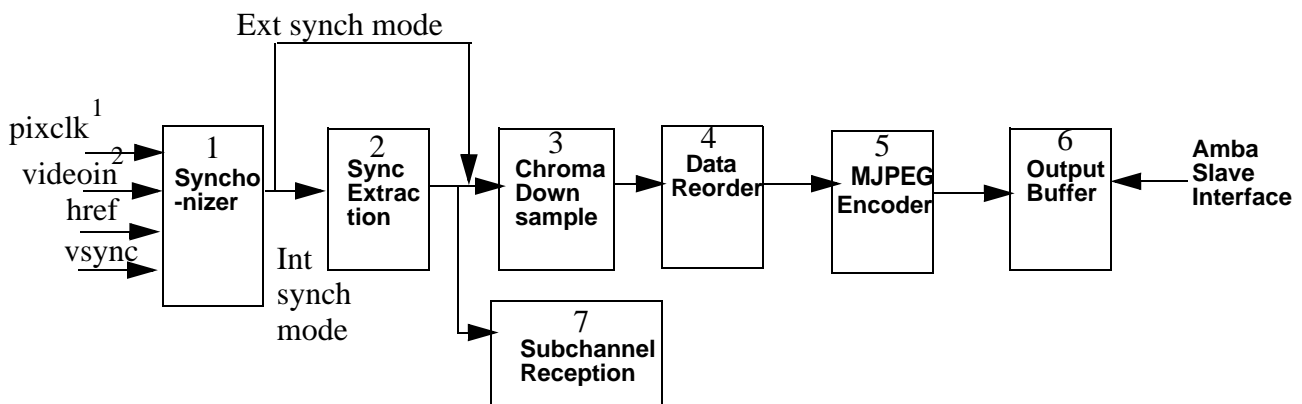
Synchronizer receives the input video stream, and changes clocking to the **ipg_video_clk** (128 MHz). If the input video stream is ITU-BT656 like compliant, it is fed to the sync extraction or decoder. The sync extraction block extracts the ITU-BT656 like sync (FF-00-00), and sends the video to the processing functions.

If the video stream is non ITU-BT656 like compliant, it does not go through the sync extraction block and is directly fed to the third block.

Block 3 converts the 4:2:2 stream to a 4:2:0 stream by providing interpolation on the chroma components of the stream.

Block 4 performs the reordering of the stream from scan line order to MCU block order.

After this, the stream is MJPEG encoded. The encoded stream is then written to a circular buffer SRAM.



1. *pixclk* is 80–96 MHz.
2. *videoin* data is YUV422 or ITU656 compliant stream.

Figure 218. Video encoder block diagram

The block has the capability to send interrupt on the following conditions:

- Start of frame
- End of picture
- Pixel count mismatch in a line
- Line count error in a frame
- Protection bit error in decoder
- JPEG In stream received from JPEG SRAM
- Subchannel received
- Buffer filling alarm
- Configuration Error (from MJPEG Encoder)

21.1.1 Features

- Baseline/extended sequential ISO/IEC 10918-1 JPEG encoder (8/12 bit)
- Programmable huffman tables (2AC, 2DC) and quantization tables (4)
- Embedded sync (ITU-BT656 like) or external synced input interface, supporting 422 video formats
- Receives 'embedded line' information from sensor, containing all register settings, as 'subchannel information' into dedicated SRAM.
- Encoded stream is stored in SRAM output buffer memory, accessible over slave AHB bus.

21.2 Block Diagram

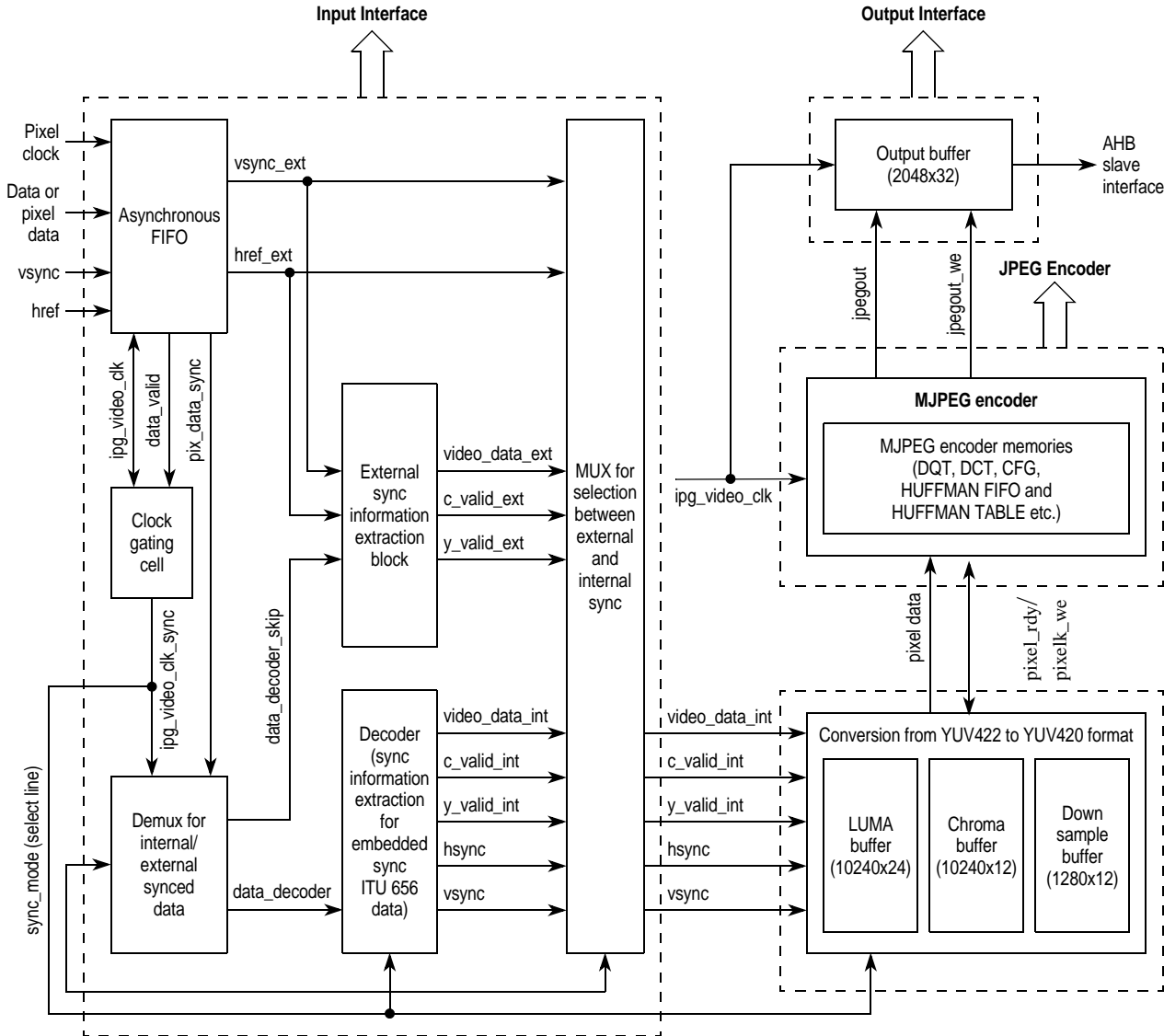


Figure 219. Video Encoder Wrapper Block Diagram

21.2.1 MJPEG Video Encoder

The MJPEG encoder is a third party IP, which performs 8/12 bit data encoding. The video encoder needs to be programmed using the MJPEG encoder configuration registers. All the MJPEG control/status registers are available for configuration through the Video Encoder Wrapper. The encoder will output compressed data to the circular/output buffer which is an AHB slave to read by the Ethernet/DMA.

The MJPEG Encoder needs to be pre-configured to define the Huffman & Quantization tables, frame format, length of restart interval etc. This information is provided to the MJPEG IP using its configuration Interface.

To implement this, the video encoder wrapper has a JPEGIn Buffer (256x32 bits). This buffer RAM needs to be configured with the configuration stream prior to starting the MJPEG encoder. The RAM is accessible through the IPS interface for writing. During the vertical blanking period, the core can configure this RAM with the values required. The Video Encoder Wrapper also specifies JPEGIn offset address register, which specifies the offset in the RAM from where the reading of configuration data shall start. The first byte is taken from the address pointed to by JPEGIn Offset address register. Subsequent data bytes are from subsequent addresses. The configuration stream is terminated using the EOI marker. Then the MJPEG is configured to read in the configuration stream from the JPEGIn buffer. After the MJPEG Encoder reads the configuration stream, the *Jpeg IN stream IRQ* interrupt is generated. This interrupt can be cleared by writing one to the clear bit.

21.2.2 MJPEG Operation Modes

Figure 220 shows the control flow implemented by the JPEG encoder.

After reset, the JPEG encoder enters idle mode. In the idle mode, access to the control and the status registers is enabled. The MJPEG encoder exits the idle mode when the CONF or GO bit of the control register are detected to be 1. The core then enters one of the operation's mode. See Section 21.3.2.12, "MODE".

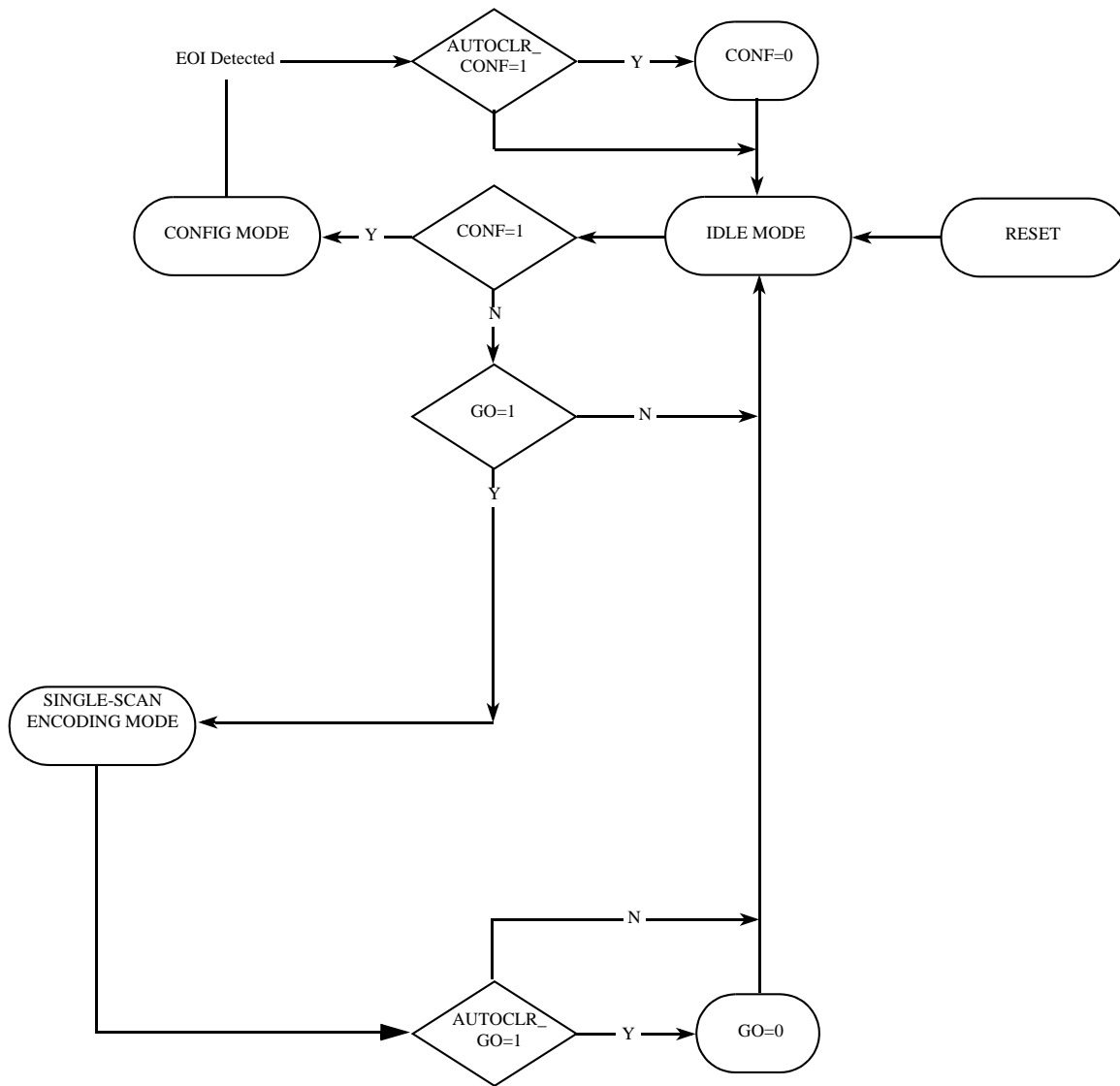


Figure 220. Global control flow (LP=SWR=0)

21.2.2.1 Configuration Mode

Configuration mode is entered after the idle mode, if the CONF bit of the MODE control-register is detected to be 1. During the configuration mode, the core receives marker segments through the JPEGIn interface.

Marker segments define the following:

- Huffman and Quantization tables (DQT and DHT marker segments)
- Frame format (SOF0)
- Length of the restart interval (DRI)

- Start of Scan (SOS)
- Comment and application marker segments (COM and APP)

The DQT, DHT, SOF0, DRI and SOS marker segments are decoded and used for the self-configuration of the core, while the COM and APP marker segments are internally stored so that they are available for extracting in the JPEG stream whenever needed. Once configuration mode is completed, and if the AUTOCLR_CONF bit of the CFG_MODE control register is 1, the CONF bit of the MODE control register is automatically cleared.

If an error (illegal marker or corrupted marker segment) is detected during marker segments' decoding, the error is reported in the [JPEG Stat 12 field description](#) and the ConfigError output pin is asserted. In order to continue working properly, a Software Reset command should be given via the control-register.

NOTE

After Soft reset (bit 30 in [Status_config](#) register) is asserted and released, CAST registers need re-configuration. The wrapper registers need not be re-configured once soft-reset is released, but all CAST configuration registers like MODE, CFG_MODE, and so on need to be re-configured for proper re-functioning of the video encoder sub-system.

The referenced marker segments must be packed in a configuration-stream. A configuration stream must always start with a SOI marker and must always be terminated with an EOI marker. The configuration stream may contain the rest of the marker segments reported in the table below.

Marker	Code	Description
SOI	FFD8h	Start of image
DQT	FFDBh	Define Quantization table(s)
DHT	FFC4h	Define Huffman table(s)
DRI	FFDDh	Define restart interval
SOF0	FFC0h	Baseline frame definition
SOF1	FFC1h	Extended Sequential frame definition
SOS	FFDAh	Start of scan
COM	FFFEh	Comment
APPn	FFE0h-FFEFh	Application segment, n=0...F

The configuration mode must be entered in one of the following cases:

1. After to perform encoding. In this case, the format of the configuration stream must be as follows:

SOI (Start Of Image)

APPn (Application Segment: Optional)

COM (Comment segment: Optional)

DQT (Quantization Table(s) segment(s))
 SOF0 (Start Of Frame (JPEG Baseline) segment) (EXTSEQ=0) or
 SOF1 (Start Of Frame (JPEG Extended) segment) (EXTSEQ=1)
 DHT (Huffman Table(s) segment(s))
 DRI (Restart Interval segment: Optional)
 SOS (Start Of Scan segment(s))

EOI (End Of Image)

- Whenever the frame format and/or the encoding options, as defined by the most recently decoded DHT, DQT, SOF0, DRI and SOS marker segments, will change for the next frame or scan to be encoded. In those cases, the configuration stream needs to contain only the marker segments that need to be updated. So, for example if only the DQT marker needs to be updated, the configuration stream can be:

SOI (Start Of Image)

DQT (Quantization Tables(s) segment(s))

EOI (End Of Image)

NOTE

Whenever the SOF0 is updated, the SOS marker must be updated too.

21.2.2.2 Encoding Mode

The MJPEG encoder will enter the single-scan encoding mode, if the control-register is configured so that: CONF=0 and GO=1. As in all encoding modes, the core receives image samples on an MCU, raster scan order via the Pixel-In interface and outputs a Baseline ISO/IEC 10918-1 JPEG stream via the Jpeg-Out interface.

[Figure 221](#) shows the control flow implemented under single-scan encoding mode. Specifically, the core automatically outputs on the Jpeg-Out interface the marker segments that are not masked by the bits 0-7 of the control register (see [Table 227](#)). It then encodes the frame samples and outputs the entropy-coded segments via the Jpeg-Out interface. See [Section Table 239., “CFG_MODE field description”](#). After the entire scan is encoded the core extracts the EOI marker and leaves the encoding mode.

To identify the end of an image (scan), the SOF0 (0 need to be an index) marker needs to define the image geometry (Y-field > 0). Thus, the MJPEG Encoder apriori knows the number of samples to expect in each scan and it just counts the incoming samples.

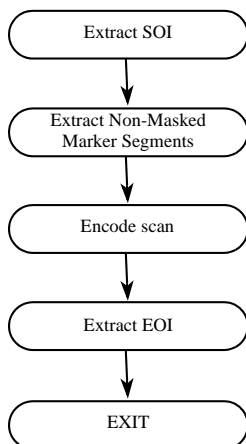


Figure 221. Control flow under single-scan encoding mode

21.2.2.3 Rate Control operation

In principle there are two different set of methods of controlling the bit rate of a jpeg stream. One set of methods is about controlling the output data peak bandwidth, and the other one is about controlling the total size of the jpeg output stream. The first set of methods applies to bandwidth limited applications, while the second one applies to storage limited applications.

The size based rate control gives superior results in terms of image quality but it can guarantee maximum output rate only on a per frame basis. The size control set of methods have the add-on basic principle of accumulating “unused threshold bits” from past DCT blocks (this happens in areas of the image which are highly compressed) and then using these bits in the more demanding image areas, compensating thus the quality around the whole image. Size based rate control needs to be done using software.

This version of the JPEG-E-X does not implement size based rate control; rather it implements bandwidth based rate control.

The bandwidth based rate control is activated when one or both of the LUMTH and CHRTH control registers are programmed with a value different than FFFF. This option of the implemented rate control is to set a maximum number-of-bits threshold which applies uniformly to all DCT blocks. During encoding each DCT block is prevented from producing more encoded bits than this threshold by zeroing out as many as needed, if needed at all, quantized DCT coefficients in the Zig-Zag scan order.

One optimization of this method, implemented as a second option, is to use different thresholds between Luminance and Chrominance DCT blocks. The core supports this by programming different values to the LUMTH and CHRTH control registers. This option gives better results, in terms of image quality, since the user can allocate different bandwidth between Luminance and Chrominance data. LUMTH and CHRTH needs to be selected according to the read bandwidth of the output buffer. Thus, an overflow of the circular output buffer can be avoided.

NOTE

In all cases proper initial selection of the quantization tables is a critical factor for the resulting image quality after rate control.

21.3 Memory Map and Register Definition

21.3.1 Memory Map

Table 227. Video Encoder memory map

Offset or Address	Video Encoder registers	Access	Section/Page
0x00 ¹	Status_config	RW	21.3.2.1/457
0x04 ¹	Picture_size	RW	21.3.2.2/459
0x08 ¹	Pixel_count	RW	21.3.2.3/460
0x0c ¹	Reserved		
0x10 ¹	Dma_address	R	21.3.2.4/460
0x14 ¹	Dma_vstart_address	RW	21.3.2.5/461
0x18 ¹	Dma_vend_address	R	21.3.2.6/461
0x1C ¹	Dma_alarm_address	RW	21.3.2.7/462
0x20 ¹	Subchannel buffer start	RW	21.3.2.8/462
0x24 ¹	Jpeg in offset address	RW	21.3.2.9/463
Control registers			
0x28	RC_REGS_SEL	W	21.3.2.10/463
0x2c	LUMTH	W	21.3.2.11/464
0x30 ²	MODE	W	21.3.2.12/465
0x34	CFG_MODE	W	21.3.2.13/466
0x38	CHRTH	W	21.3.2.14/467
Status registers			
0x40	JPEG stat 0	R	21.3.2.16/468
0x44	JPEG stat 1	R	21.3.2.17/468
0x48	JPEG stat 2	R	21.3.2.18/469
0x4C	JPEG stat 3	R	21.3.2.19/469
0x50	JPEG stat 4	R	21.3.2.20/470
0x54	JPEG stat 5	R	21.3.2.21/470
0x58	JPEG stat 6	R	21.3.2.22/471
0x5C	JPEG stat 7 (Not Used)	R	21.3.2.23/471
0x60	JPEG stat 8	R	21.3.2.24/472
0x64	JPEG stat 9	R	21.3.2.25/472
0x68	JPEG stat 10	R	21.3.2.26/473
0x6C	JPEG stat 11	R	21.3.2.27/473

Table 227. Video Encoder memory map (continued)

Offset or Address	Video Encoder registers	Access	Section/Page
0x70	JPEG stat 12	R	21.3.2.28/474
0x74	JPEG stat 13	R	21.3.2.29/475
0x78	JPEG stat 14	R	21.3.2.30/475
0x7c	JPEG stat 15	R	21.3.2.31/476
0x0200–0x02FF	Subchannel buffer space (256 bytes)	RW	—
0x1000–0x13ff	Jpeg in buffer space (1024 bytes)	RW	—
0x5000_0000– 0x5000_3FFF	circular buffer space. This does not reflect in IPS address space. Instead this is AHB mapped address range.	RW	—

¹ The registers from offset 0x00 to 0x24 remains accessible if Video Encoder peripheral is frozen through ME_PCTL30 register in all the modes.

² Registers 0x28 - 0x7c are resident in the MJPEG encoder but are mapped in Wrapper. The registers 0x28-0x38 are Write Only through wrapper. We cannot read the reset values of these registers as well. However JPEG_encode CFG_MODE register can be read from address 0x74. LUMTH can be read from 0x78. CHRTH can be read from 0x7c

NOTE

Video output buffer RAM inside Video Encoder should not be used as a General Purpose system RAM.

21.3.2 Register Descriptions

21.3.2.1 Status_config

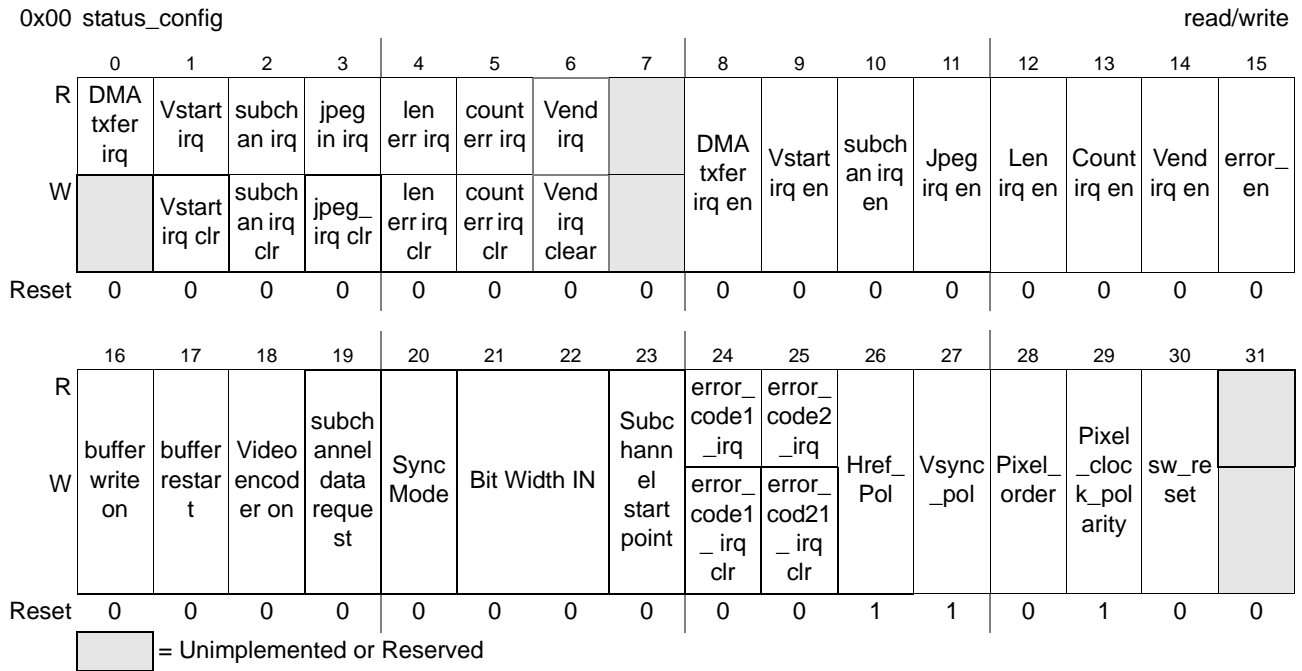


Figure 222. status_config Register

Table 228. status_config Fields

Field	Description
DMA txfer irq	DMA transfer request - given every time dma_address >= dma_alarm_address
Vstart irq	Interrupt signalling the SOI marker of the MJPEG Encoder i.e. the MJPEG is configured for encoding.
Subchannel data irq	Subchannel data received and ready for processor read.
Jpeg IN stream irq	Request for JPEG In stream received from MJPEG encoder, and data provided from JPEGIn RAM.
Line Len err irq	Interrupt signalling mismatch between active line length and programmed line length
Line count err irq	Interrupt signalling mismatch between active number of lines in an image and programmed number of lines in the image
Vend irq	Interrupt signalling the completion of encoding of Current Frame. It is asserted at the detection of EOI
DMA txfer irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
Vstart irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.

Table 228. status_config Fields

Field	Description
Subchannel data irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
Jpeg IN stream irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
Line Len err irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
Line count err irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
Vend irq_en	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
Error En	1'b1 : Interrupt is Enabled. 1'b0: Interrupt is Disabled.
buffer_write_on	1: write to buffer enabled 0: write to buffer disabled Note:- Bit will be open to be written by the software all the time, but active state of bit will be locked at the SOI marker of the frame, and will be open to be changed again at the end of current frame encoding.
buffer_restart	1: dma_address will reload from dma_vstart_address at next SOI 0: dma_address will free-run, is not updated on next SOI Bit clears automatically on dma_address reload.
Video_encoder_on	1: turn video encoder on 0: video encoder off Note:- Bit will be open to be written by the software all the time, but active state of state of bit will be locked at the positive edge of vsync, and will be open to be changed again at the end of current frame encoding.
Subchannel data request	1: Request video in block to receive subchannel data 0: Do not request subchannel data. Bit will auto-clear on reception of subchannel data. After receiving subchannel data, Subchannel Data IRQ will be generated. If no data is requested, interrupt is not generated.
Sync Mode	0: Hsync/Vsync external signals used for syncing 1: ITU-BT656-like embedded syncs
Bit Width In	00 : 8 bit 01 : 10 bit 10 : 12 bit 11 : Reserved, do not use
subchannel start point	0: start counting pixels from the starting edge of vsync 1: start counting pixels from first valid pixel of frame
error code1 irq	Interrupt signalling protection single bit error in decoder
error code2 irq	Interrupt signalling protection double bit error in decoder
href_pol	Defines the active polarity of HREF signal.Applicable only for External Sync Mode 1'b1: HREF is active high 1'b0: HREF is active low

Table 228. status_config Fields

Field	Description
vsync_pol	Defines the active polarity of VSYNC signal. Applicable only for External Sync Mode 1'b1: VSYNC is active high 1'b0: VSYNC is active low
pixel_order	Defines whether first chroma pixel or luma pixel will come in the camera input stream. Applicable both for internal & external sync modes. 1'b1: Chroma pixel first in YCbCr Stream 1'b0: Luma pixel first in YCbCr stream.
pixel_clock_pol	Defines whether data is sampled on positive or negative edge of pixel clock 1'b1: data is sampled on positive edge of pixel clock 1'b0: data is sampled on negative edge of pixel clock
sw_rst	Initialises all flops of design on to a known state. Signal is active high. Note: This bit must be cleared to exit reset state.

21.3.2.2 Picture_size

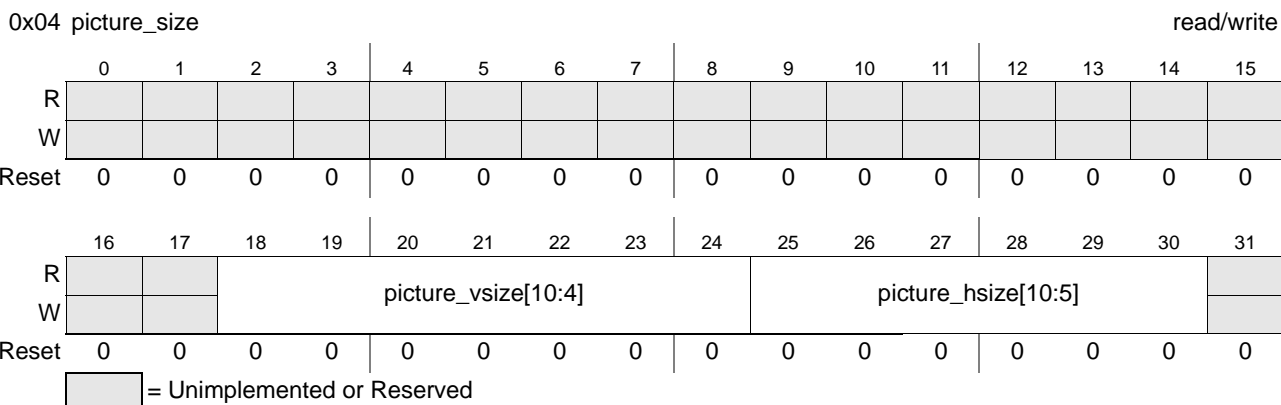


Table 229. Picture size register fields

Field	Description
picture_hsize[10:5]	Number of pixels in line, increments of 32. For example, if hsize is 256 then the value to be programmed is 'b001000. Number of pixels in a line should be divided by 32. You must carefully provide number of lines in a pixel as multiples of 32.
picture_vsize[10:4]	Number of lines in field, increments of 16.

21.3.2.3 Pixel count

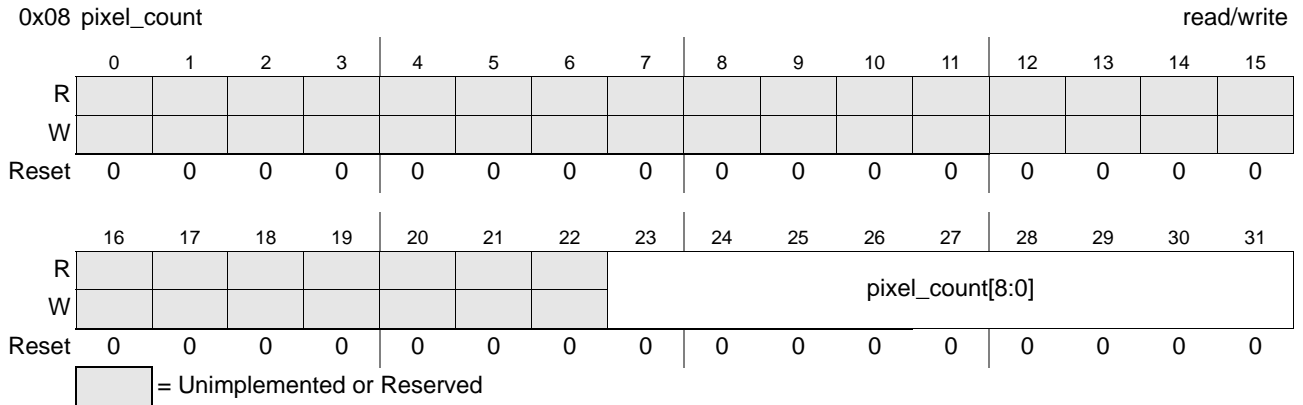


Table 230. Pixel count register fields

Field	Description
pixel_count[8:0]	Number of pixels to be stored in subchannel RAM.

21.3.2.4 Dma_address

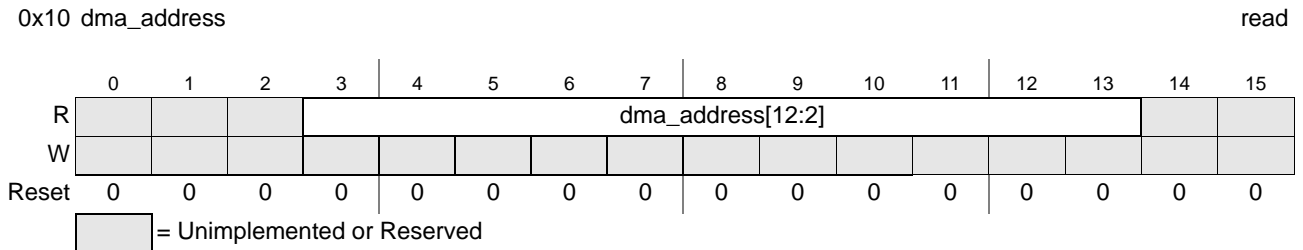


Figure 223. dma_address Register

Table 231. dma_address Fields

Field	Description
dma_address[12:2]	Address where the video encoder is currently writing in the output buffer. This can indicate the current buffer fill level.

21.3.2.5 Dma_vstart_address

0x14 dma_vstart_address

read/write

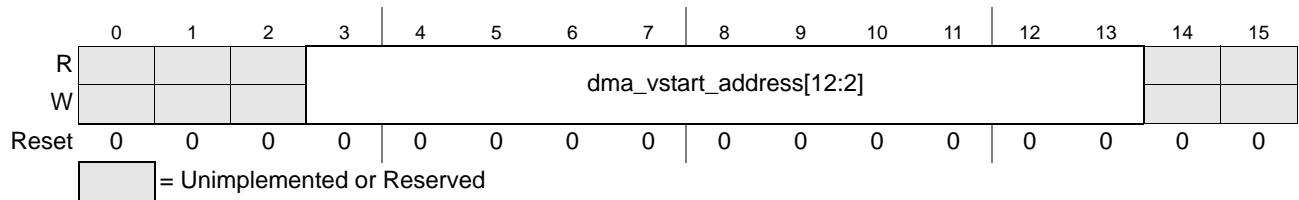


Figure 224. dma_vstart_address Register

Table 232. dma_vstart_address Fields

Field	Description
dma_vstart_addresses[12:2]	Address where the write of next frame will start if buffer_restart bit is 1'b1.

21.3.2.6 Dma_vend_address

0x18 dma_vend_address

read

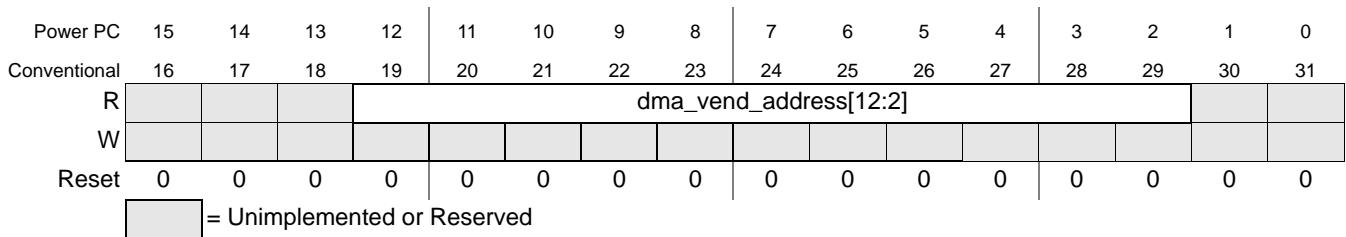


Figure 225. dma_vend_address Register

Table 233. dma_vend_address Fields

Field	Description
dma_vend_addresses[12:2]	Belongs with vend interrupt. Address at which write of current frame ended.

21.3.2.7 Dma_alarm_address

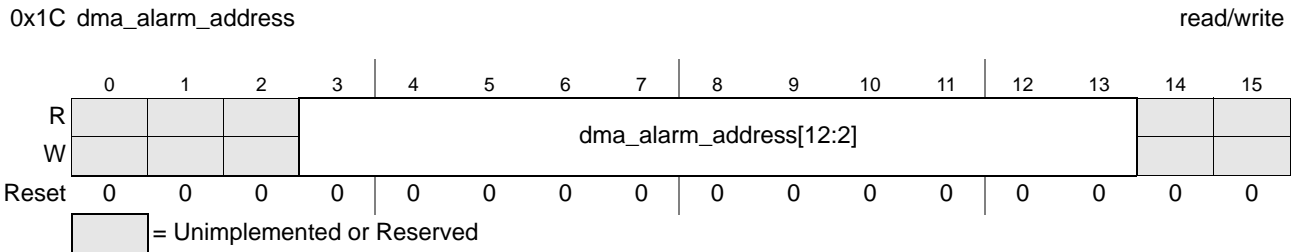
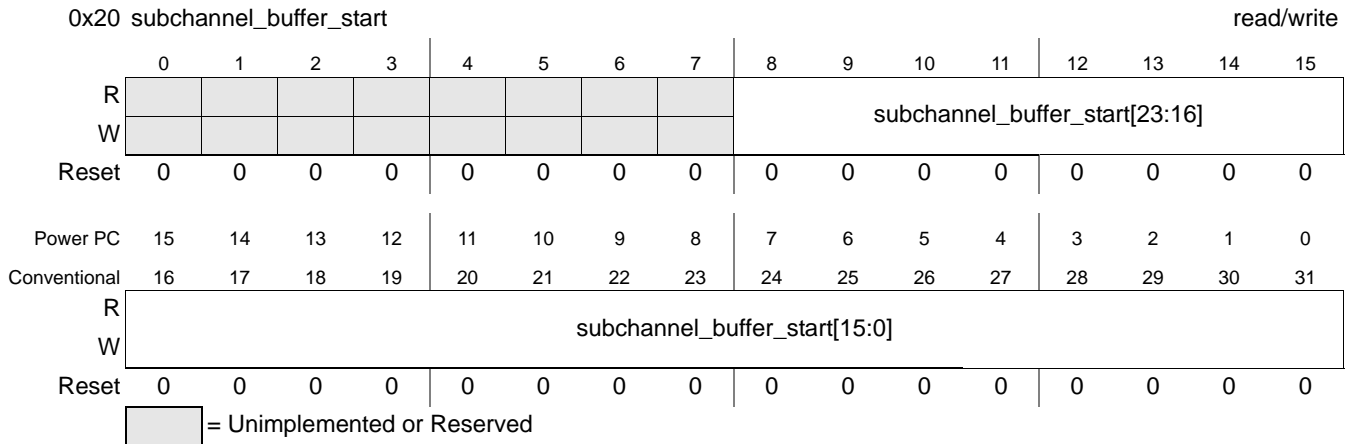


Figure 226. dma_alarm_address Register

Table 234. dma_alarm_address Fields

Field	Description
dma_alarm_address[12:2]	When dma_address >= dma_alarm_address, alarm interrupt is generated.

21.3.2.8 Subchannel buffer start



Field	Description
subchannel_buffer_start[23:0]	Subchannel buffer start address. This specifies the number of pixel clocks after HREF/VSYNC from where subchannel data starts arriving from camera.

21.3.2.9 JPEG In Offset Address

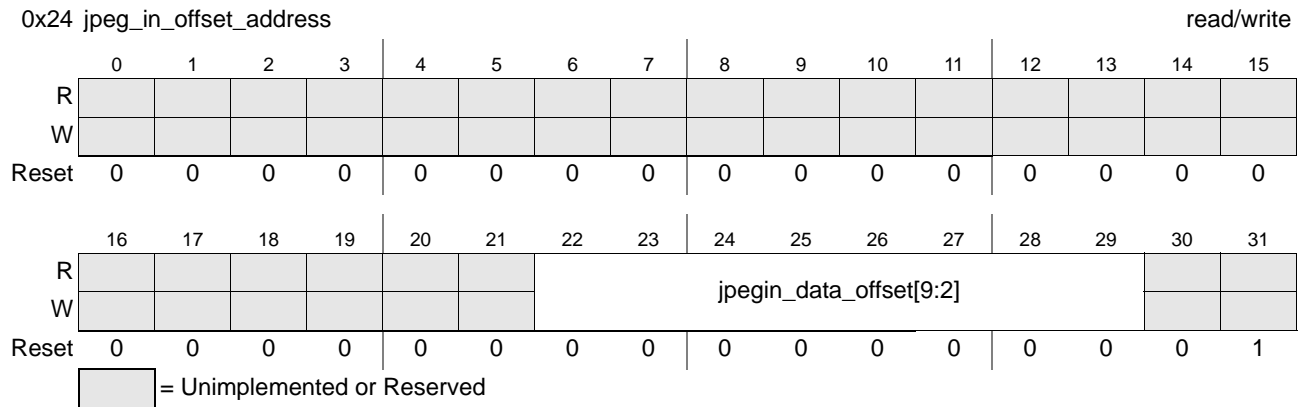


Table 235. JPEG data in offset Fields

Field	Description
jpeg data in offset[9:2]	This is a pointer in JPEG in buffer RAM where jpeg IN stream will be sourced next time MJPEG encoder requests

21.3.2.10 RC_REGS_SEL

The RC_REGS_SEL control register is used as an indirect status register select. It selects between the two sets of Rate-Control (RC) status registers, which are available for read at JPEG stat 14 and JPEG stat 15. The RC_REGS_SEL register can be programmed with the values 0, 1 and 2. The corresponding set of registers that will be available for read through status registers 14 and 15 is shown in the [Table 236](#):

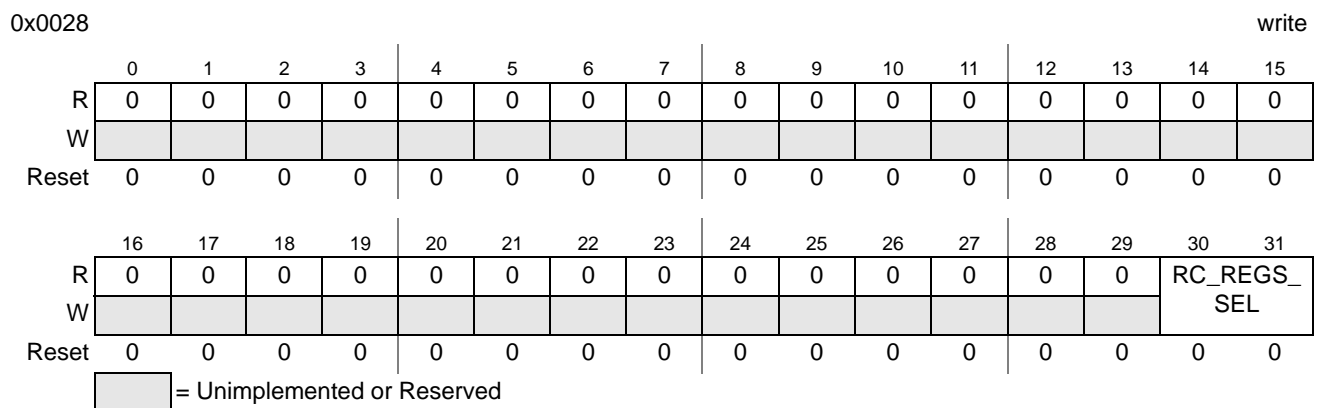


Figure 227. RC_REGS_SEL control register

Table 236. The RC_REGS_SEL control register

RC_REGS_SEL Value	RC_REGS0 (JPEG stat 14)	RC_REGS1 (JPEG stat 15)
0	LUMATH	CHROMATH
1	LumaTruncH (Bits 31:16 of register with total Truncated Bits of Luminance blocks).	LumaTruncL (Bits 15:0 of register with total Truncated Bits of Luminance blocks).
2	ChromaTruncH (Bits 31:16 of register with total Truncated Bits of Chrominance blocks).	ChromaTruncL (Bits 15:0 of register with total Truncated Bits of Chrominance blocks).

21.3.2.11 LUMTH

This register is used with AC(0) Huffman Table. Huffman Tables must be assigned for Luminance components in SOS marker segment accordingly. Maximum allowed programmed threshold value for both registers is 0x03DF.

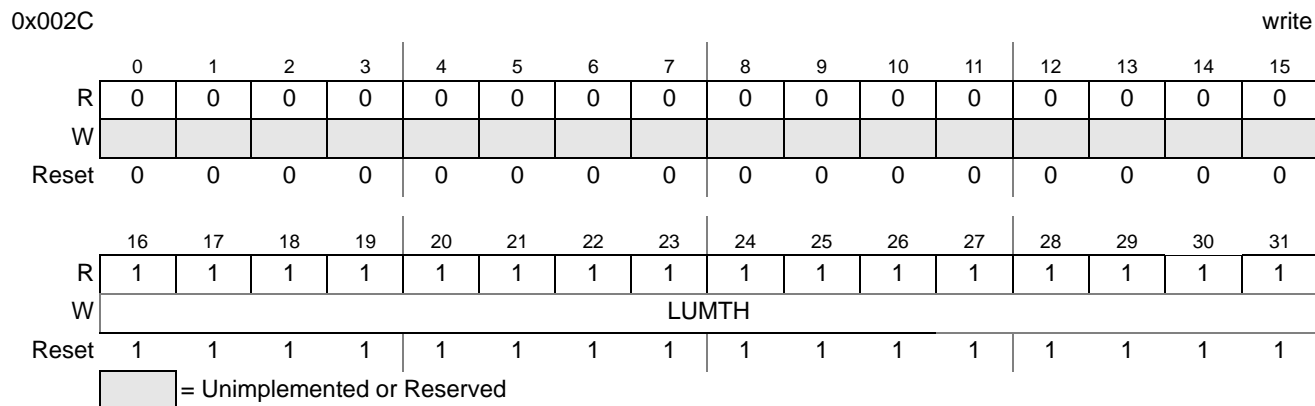


Figure 228. LUMTH register

Table 237. LUMTH field description

Field	Description
LUMTH	Maximum number of bits threshold used in rate control of Luminance DCT blocks (the maximum value is 0x03DF).

21.3.2.12 MODE

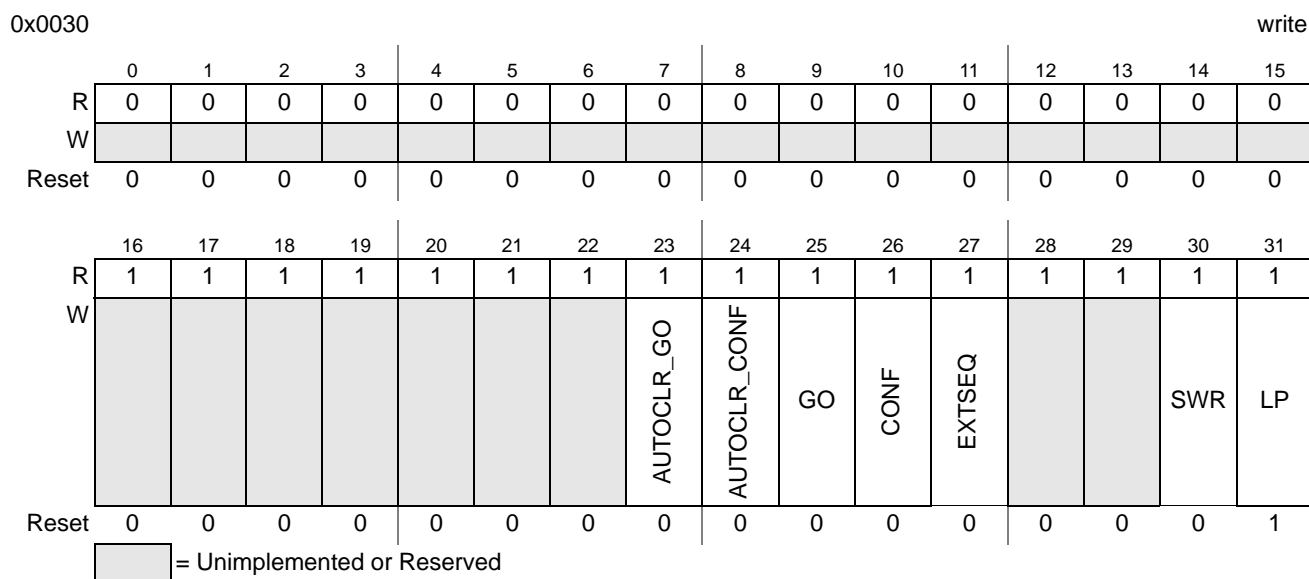


Figure 229. MODE register

Table 238. MODE field description

Field	Description
LP	Low Power 0 LP is disabled 1 The core enters low-power mode (all internal registers are frozen)
SWR	Soft Reset 0 SWR is not enabled 1 The core can be reset
EXTSEQ	Selector between Baseline and Extended Sequential mode of operation 0 Baseline mode is selected 1 Extended Sequential mode
CONF	Configuration. When 1 the core enters configuration mode
GO	GO 0 GO is disabled 1 The core exits the idle mode
AUTOCLR_CONF	Auto clear CONF bit when the core exits from configuration mode
AUTOCLR_GO	Auto clear GO bit when the core exits from its current encoding mode of operation

21.3.2.13 CFG_MODE

0x0034 write¹

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
W						DICOM	COMB_DHT	COMB_DQT	MCOM	MAPP	MDNL	MSOS	MDHT	MDQT	MDRI	MISOFO
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

= Unimplemented or Reserved

Figure 230. CFG_MODE register

¹ The register is Write only through wrapper. The reset values of this register cannot be read. However, the register can be read from the address 0x74.

Table 239. CFG_MODE field description

Field	Description
MISOFO	Mask Start of Frame 0 No SOF0 segments are output by the core 1 SoF0 segments are output by the core
MDRI	Mask DRI 0 No DRI segments are output by the core 1 DRI segments are output by the core.
MDQT	Mask DQT 0 No DQT segments are output by the core 1 DQT segments are output by the core
MDHT	Mask DHT 0 No DHT segments are output by the core 1 DHT segments are output by the core
MSOS	Mask SOS 0 No SOS segments are output by the core 1 SOS segments are output by the core
MDNL	Mask DNL 0 No DNL segments are output by the core 1 DNL segments are output by the core
MAPP	Mask APP 0 No APP segments are output by the core 1 APP segments are output by the core

Table 239. CFG_MODE field description

Field	Description
MCOM	Mask COM 0 No COM segments are output by the core 1 COM segments are output by the core
COMB_DQT	0 Transmits all Quantization Tables in one combined DQT segment to support EXIF 2.2 format 1 Do not transmit all Quantization Tables in one combined DQT segment to support EXIF 2.2 format
COMB_DHT	0 Transmits all Huffman Tables in one combined DHT segment to support EXIF 2.2 format 1 Do not transmits all Huffman Tables in one combined DHT segment to support EXIF 2.2 format
DICOM	0 Transmits SOF0 and SOS markers after DQT and DHT markers to support DICOM format 1 Do not transmits SOF0 and SOS markers after DQT and DHT markers to support DICOM format

21.3.2.14 CHRTH

This register is used with AC(1) Huffman Table. Huffman Tables must be assigned for Chrominance components in SOS marker segment accordingly. Maximum allowed programmed threshold value for both registers is 0x03DF.

0x0038 write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
W	CHRTH															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

= Unimplemented or Reserved

Table 240. CHRTH_SEL_ADDR field description

Field	Description
CHRTH	Maximum number of bits threshold used in rate control of Chrominance DCT blocks (the maximum value is 0x03DF).

21.3.2.15 Status registers

21.3.2.16 JPEG Stat 0

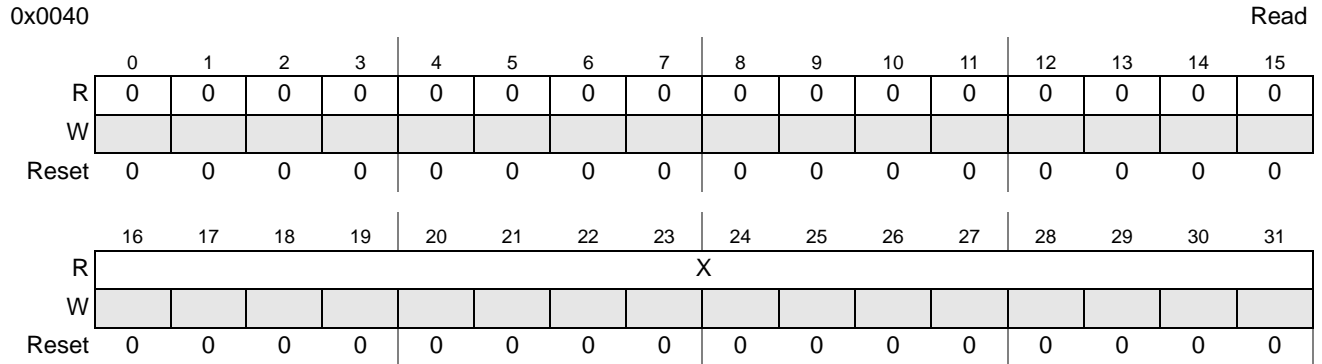


Figure 231. JPEG Stat 0 register

Table 241. JPEG STAT 0 field description

Field	Description
X	Image Width

21.3.2.17 JPEG Stat 1

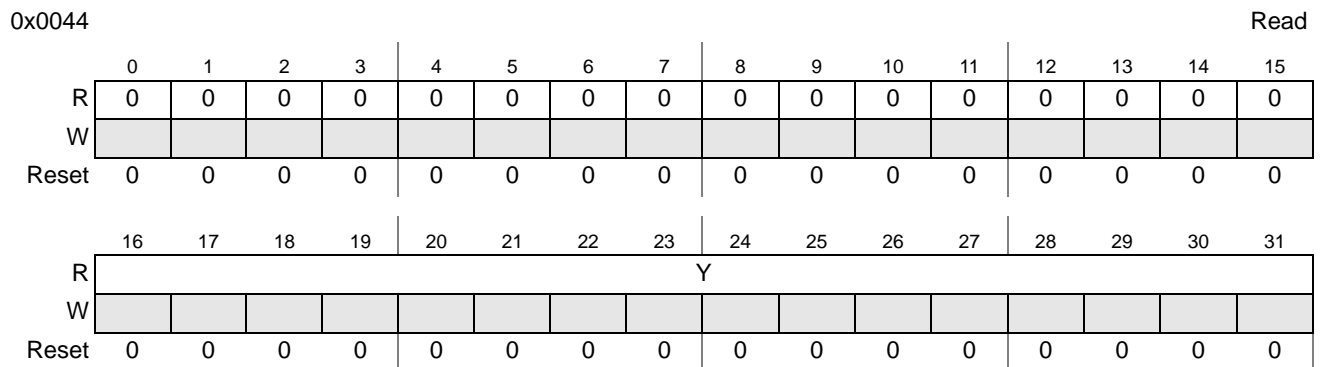


Figure 232. JPEG Stat 1 register

Table 242. JPEG Stat 1 field description

Field	Description
Y	Image Height

21.3.2.18 JPEG Stat 2

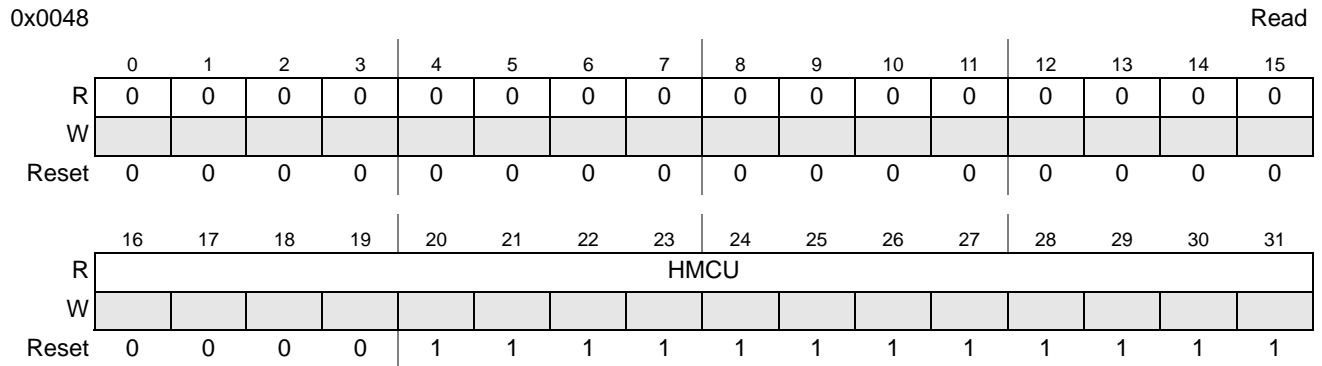


Table 243. JPEG STAT 2 field description

Field	Description
HMCU	Number of MCUs in the current scan in horizontal direction

21.3.2.19 JPEG Stat 3

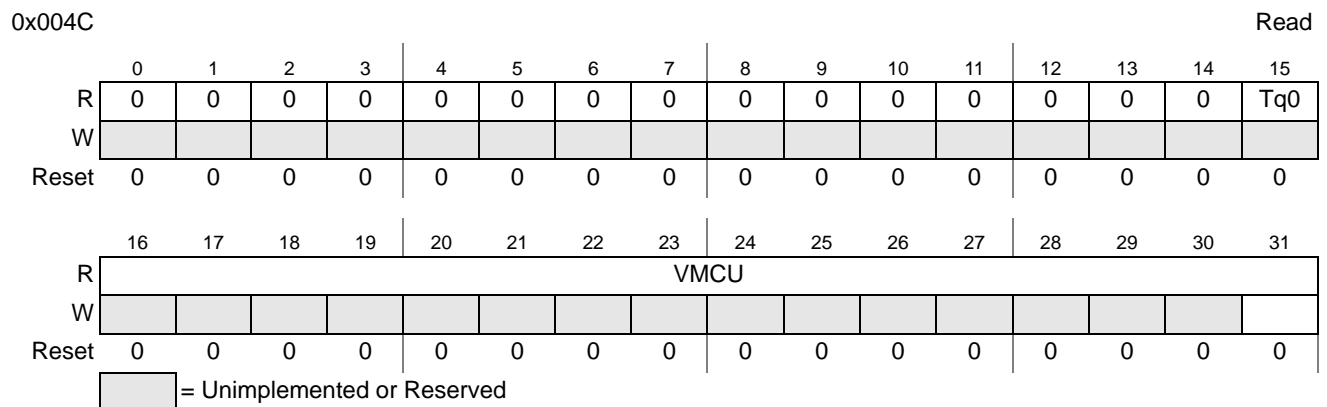


Figure 233. JPEG Stat 3

Table 244. JPEG STAT 3 field description

Field	Description
VMCU	Number of MCUs in the current scan in vertical direction

21.3.2.20 JPEG Stat 4

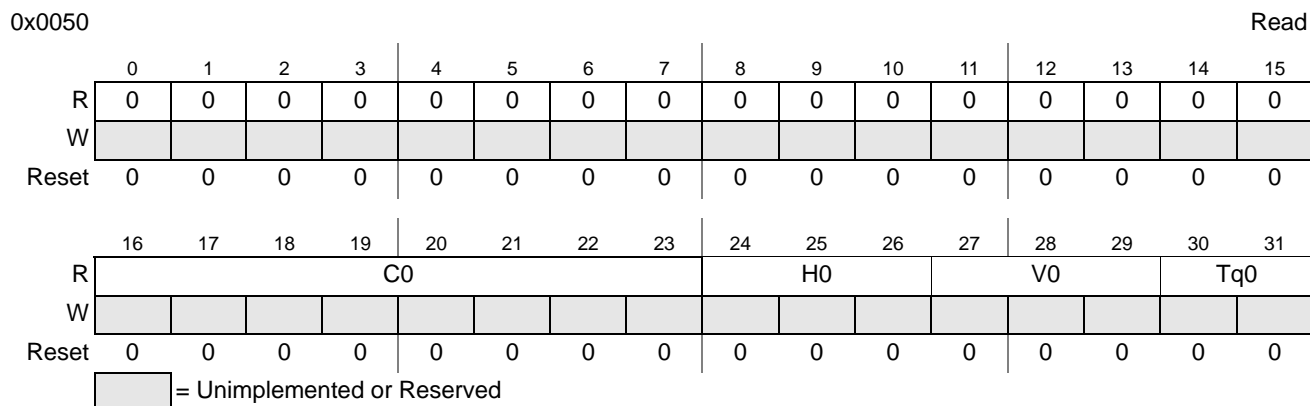


Figure 234. JPEG Stat 4 register

Table 245. JPEG Stat 4 field description

Field	Description
C0	Component identifier for scan component 0
H0	Horizontal sampling for scan component 0 (expected value for 4:2:0 = 2)
V0	Vertical sampling for scan component (expected value for 4:2:0 = 2)
Tq0	Quantization table identifier for scan component 0

21.3.2.21 JPEG Stat 5

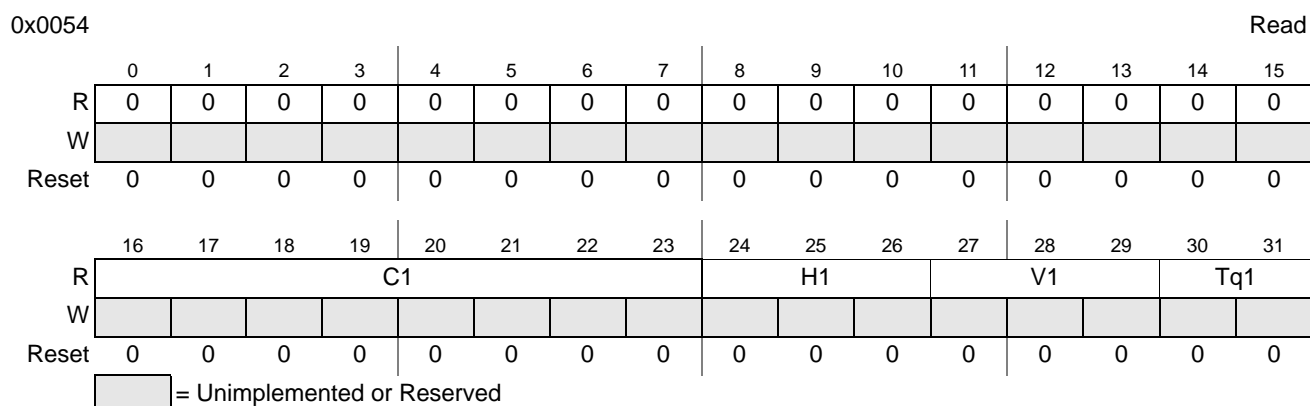
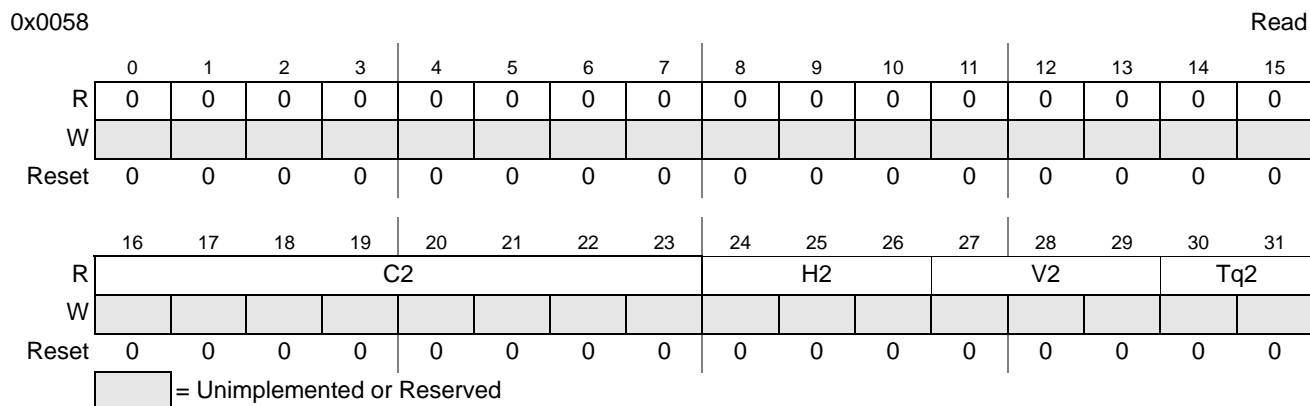


Figure 235. JPEG Stat 5 register

Table 246. JPEG Stat 5 field description

Field	Description
C1	Component identifier for scan component 1
H1	Horizontal sampling for scan component 1 (expected value for 4:2:0 = 1)
V1	Vertical sampling for scan component 1 (expected value for 4:2:0 = 1)
Tq1	Quantization table identifier for scan component 1

21.3.2.22 JPEG Stat 6


Figure 236. JPEG Stat 6 register
Table 247. JPEG Stat 6 field description

Field	Description
C2	Component identifier for scan component 2
H2	Horizontal sampling for scan component 2 (expected value for 4:2:0 = 1)
V2	Vertical sampling for scan component 2 (expected value for 4:2:0 = 1)
Tq2	Quantization table identifier for scan component 2

21.3.2.23 JPEG Stat 7

This register is not used.

21.3.2.24 JPEG Stat 8

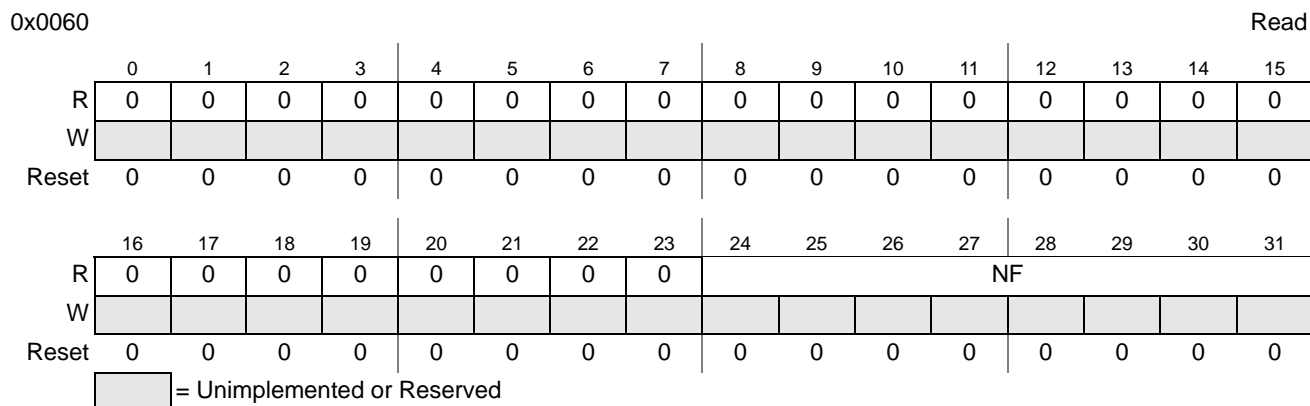


Figure 237. JPEG Stat 8 register

Table 248. JPEG Stat 8 field description

Field	Description
NF	Number of components in frame (expected value for 4:2:0 = 3)

21.3.2.25 JPEG Stat 9

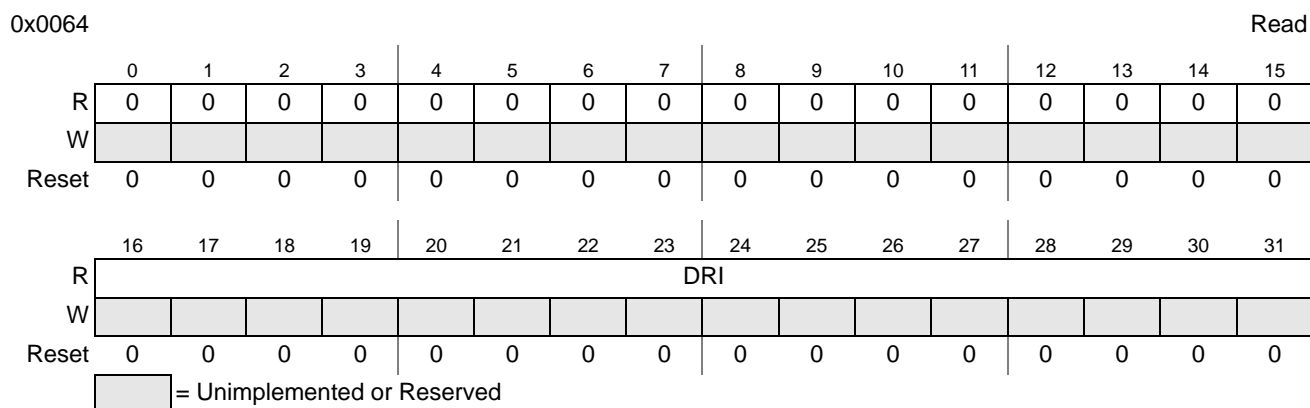


Figure 238. JPEG Stat 9 register

Table 249. JPEG Stat 9 field description

Field	Description
DRI	Restart interval

21.3.2.26 JPEG Stat 10

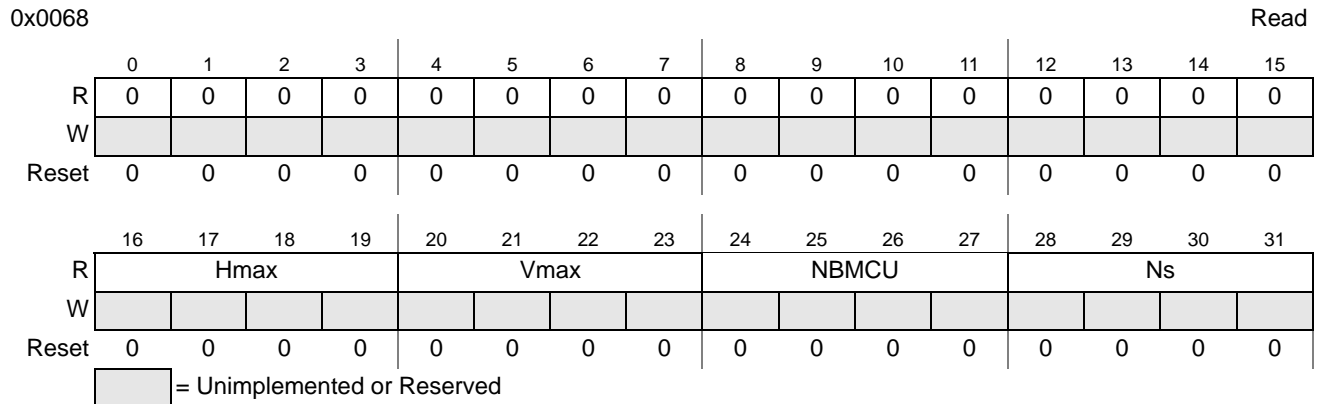


Figure 239. JPEG Stat 10 register

Table 250. JPEG Stat 10 field description

Field	Description
Hmax	Maximum horizontal sampling factor in frame (expected value for 4:2:0 = 2)
Vmax	Maximum vertical sampling factor in frame (expected value for 4:2:0 = 2)
NBMCU	Number of blocks per MCU in current scan (expected value for 4:2:0 = 6)
Ns	Number of components in current scan (expected value for 4:2:0 = 3)

21.3.2.27 JPEG Stat 11

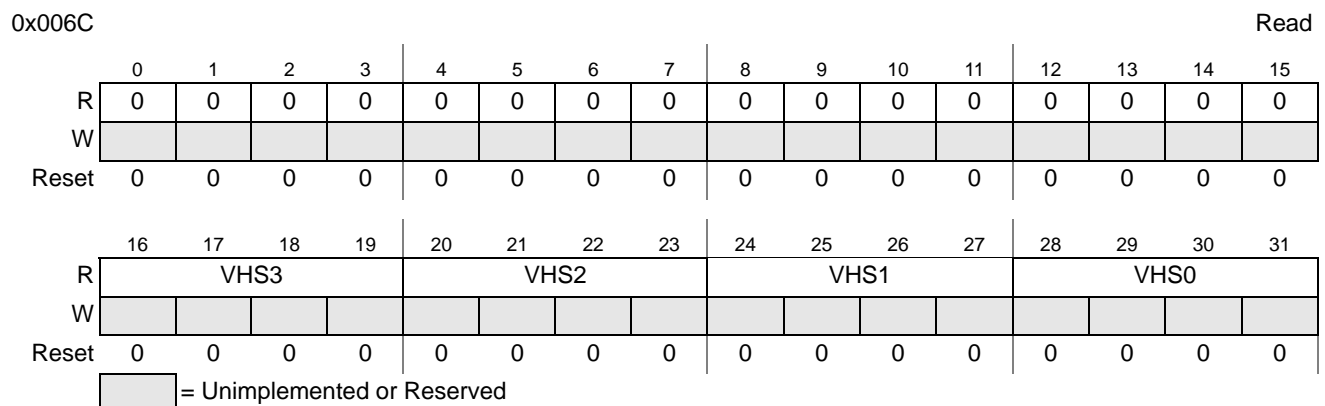


Figure 240. JPEG Stat 11 register

Table 251. JPEG Stat 11 field description

Field	Description
VHS3	Number of blocks of the fourth component in MCU. $VHS3 = VHS2 + V3 \times H3$, when $Ns = 4$ (expected value for 4:2:0 = 0)
VHS2	Number of blocks of the third component in MCU. $VHS2 = VHS1 + V2 \times H2$, when $Ns \geq 3$ (expected value for 4:2:0 = 6)
VHS1	Number of blocks of the second component in MCU. $VHS1 = VHS1 + V1 \times H1$, when $Ns \geq 2$ (expected value for 4:2:0 = 5)
VHS0	Number of blocks of the first component in MCU. $VHS0 = V0 \times H0$, when $Ns > 1$, $VHS0 = 1$ and $Ns = 1$ (expected value for 4:2:0 = 4)

21.3.2.28 JPEG Stat 12

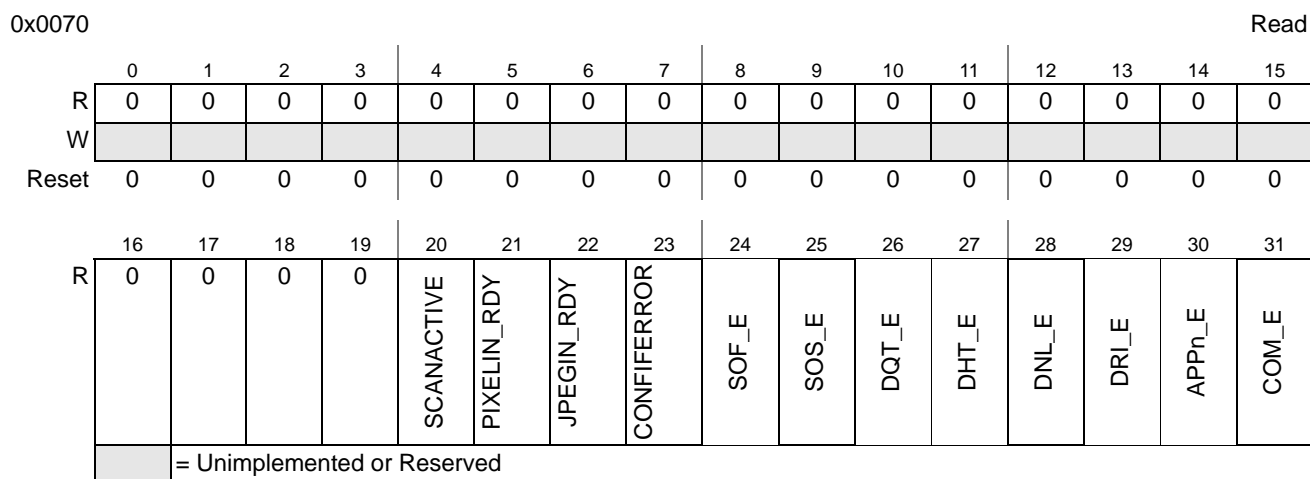


Figure 241. JPEG Stat 12 register

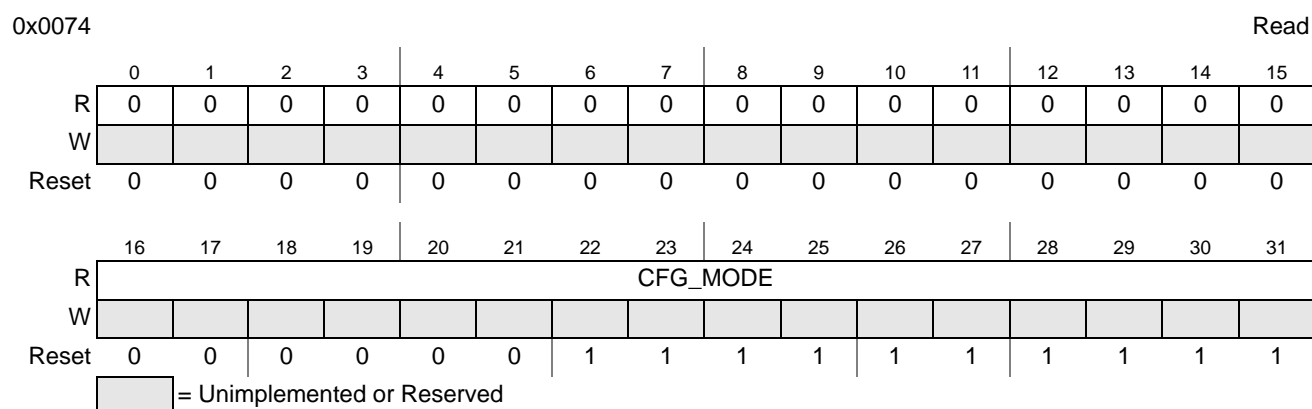
Table 252. JPEG Stat 12 field description

Field	Description
SCANACTIVE	Indicates that core encodes entropy coded scan data
PIXELIN_RDY	Pixel input data ready: Core is ready to accept new pixel data
JPEGIN_RDY	JPEG stream input data ready: Core is ready to read new data on JPEG (0x1000:0x13ff)
CONFIGERROR	Configuration error indicator
SOF_E	SOF0 error: 1 when an error in SOF0 segment is detected
SOS_E	Scan error : 1 when an error in SOS segment is detected

Table 252. JPEG Stat 12 field description

Field	Description
DQT_E	DQT error : 1 when an error in DQT segment is detected
DHT_E	DHT error : 1 when an error in DHT segment is detected
DNL_E	DNL error : 1 when an error in DNL segment is detected
DRI_E	DRI error : 1 when an error in DRI segment is detected
APPn_E	APPn error : 1 when an error in APPn segment is detected
COM_E	COM error : 1 when an error in COM segment is detected

21.3.2.29 JPEG Stat 13


Figure 242. JPEG Stat 13 register
Table 253. JPEG Stat 13 field description

Field	Description
CFG_MODE	Programmed CFG_MODE control register. Allows to read back the value programmed to 0x34.

21.3.2.30 JPEG Stat 14

This register allows to read the values as selected by RC_REGS_SEL (at offset 0x28).

0x0078 Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RC_REGS0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 243. JPEG Stat 14 register

21.3.2.31 JPEG Stat 15

This register allows to read the values as selected by RC_REGS_SEL at offset 0x28.

0x007C Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RC_REGS1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 244. JPEG Stat 15 register

21.4 Functional Description

The video encoder receives the video data, at pixel clock from camera (vsync and href also in case of external sync mode). This data is synchronised with ipg_video_clk with the help of asynchronous fifo, and then this data is made to pass through Decoder block for sync extraction(for embedded sync). The data is fed directly to the reordering logic, which downsamples the data from YUV422 to YUV420 data. This data is fed to the MJPEG encoder in MCU 8*8 Blocks (4 Luma blocks then 2 Chroma blocks), which compresses the JPEG Image and stores the data in a circular output buffer, from which the data can be read via AHB Slave Interface.

21.4.1 Input interface

The input interface accepts ITU-BT656 mode data with external href/vsync sync inputs. In case mode is set to ITU-BT656, syncing is embedded per ITU-BT656 specification in the 8 most significant data lines. In case href/vsync syncing is enabled, href and vsync signal are input. *href* is active when line pixel data is valid, *vsync* goes active during the vertical blanking. The format on the pixel data is always YUV4:2:2. Y pixels are alternated with U and V pixels.

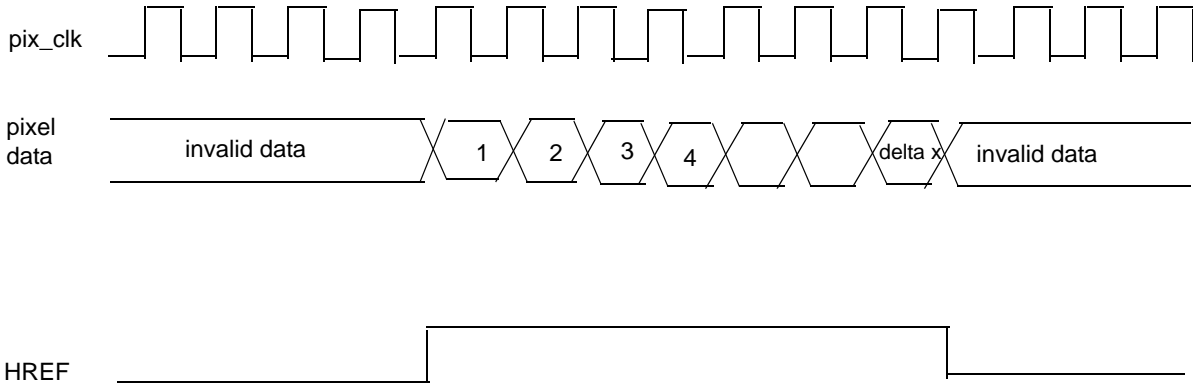
The video interface can interface with 8-bit, 10-bit or 12-bit input. Input format is 8/10/12 bit, programmable. Use of ITU-BT656-like syncing or *href/vsync* is programmable. In case 8 or 10 bit mode is used, 4 or 2 bits on LSB side of input word are not used. When less than 12 bit is taken, the data is taken from the most significant bits.

The input interface can be used in ITU-BT656-like mode, using embedded syncs, and using external *href/vsync* sync signals. Pixel clock frequency should be lower than `ipg_video_clk` at all times.

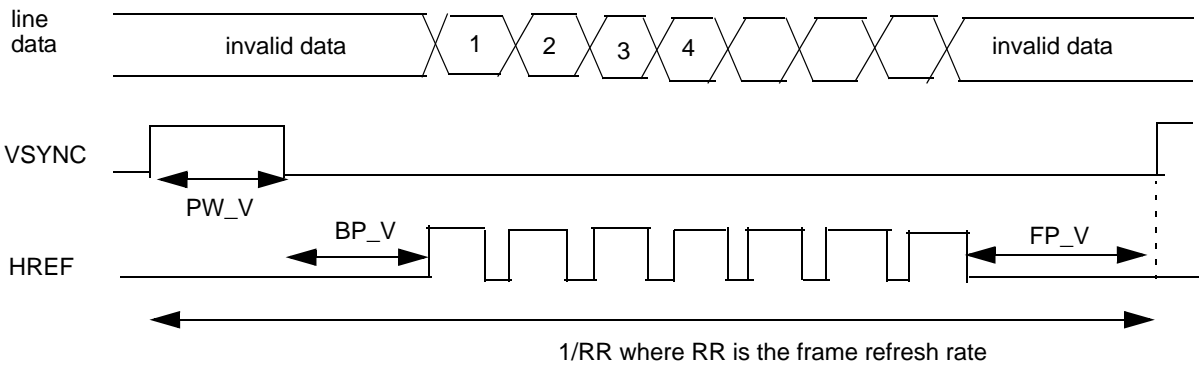
21.4.1.1 External Sync Interface Timing Diagram

[Figure 245](#) shows the timing signals from the camera. Camera sends two signals VSYNC and HREF for the input interface. The figure shows the signals for one line of data and one complete frame of data. VSYNC indicates the start of new frame while SHSYNC indicates start of new line.

[Figure 245](#) shows the timing diagram for a case when both HREF and VSYNC are active high. The design supports configurability in the active polarity for HREF, VSYNC, and `pix_clk`.



Timing Diagram for One Line of Pixel data



1. PW_V is the width of VSYNC pulse.
2. BP_V is the back porch period.
3. FP_V is the front porch period.

Note

FP_V + PW_V >= 16 lines.

Figure 245. External Sync Timing Diagram

21.4.1.2 ITU-BT656 sync information extraction

According to ITU-BT656 recommendation, the digital video input data signals will be in the form of binary signals coded in 8, 10 bit data words.

These data words can be video data signals or timing reference signals (VSYNC, HSYNC)

In this mode, the Horizontal (H), Vertical (V), and Field (F) signals are sent as an embedded part of the video data stream in a series of bytes that form a control word. The Start of Active Video (SAV) and End of Active Video (EAV) signals indicate the beginning and end of data elements to read in on each line. SAV occurs on a 1-to-0 transition of H, and EAV begins on a 0-to-1 transition of H. An entire field of video comprises Active Video + Horizontal Blanking (the space between an EAV and SAV code) and Vertical Blanking (the space where V = 1). A field of video commences on a transition of the F bit.

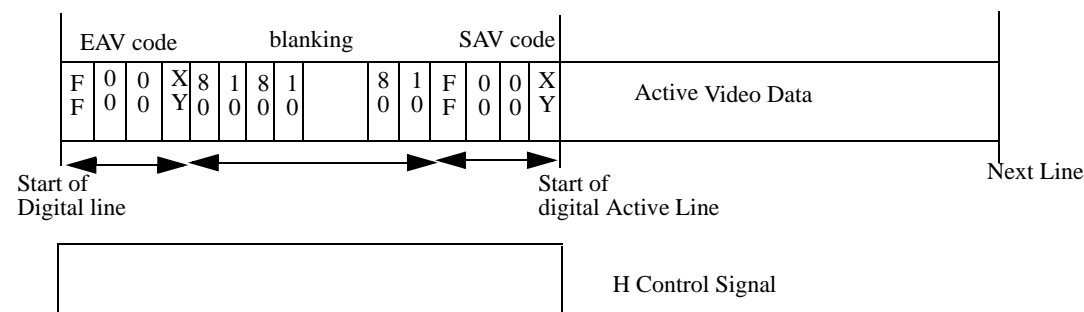


Figure 246. ITU-BT656 like 8 bit parallel data format

The SAV and EAV codes have a defined preamble of three bytes (0xFF,0x00,0x00) followed by XY status word which aside from the Field (F), Vertical blanking (V) and Horizontal blanking bits contains four protection bits for single bit error correction and detection. Also, F and V fields are only allowed to change as part of EAV sequences i.e transition from H=0 to H=1.

NOTE

- 1) 8/10 bit video is supported on this device.
- 2) Only Progressive mode is supported.

Table 254. Input data Format for Internal Sync

Data Bit	FirstWord (FF)	SecondWord (00)	ThirdWord (00)	FourthWord (XY)
D9(MSB)	1	0	0	1
D8	1	0	0	F
D7	1	0	0	V
D6	1	0	0	H
D5	1	0	0	P3
D4	1	0	0	P2
D3	1	0	0	P1
D2	1	0	0	P0
D1	1	0	0	0
D0(LSB)	1	0	0	0

The bit definitions for the status word XY are:

F = 0 for field 0

F = 1 for field 1

$V = 1$ during vertical blanking period

$V = 0$ when not in vertical blanking

$H = 0$ at SAV

$H = 1$ at EAV

$P3 = V \text{ XOR } H$

$P2 = F \text{ XOR } H$

$P1 = F \text{ XOR } V$

$P0 = F \text{ XOR } V \text{ XOR } H$

21.4.1.3 Video In Data Format for embedded Sync Mode

Physical Interface available as Input data width (pdi_datain) is 8/10/12 bits

XY word in the input stream is used to decode the value of VSYNC & HSYNC

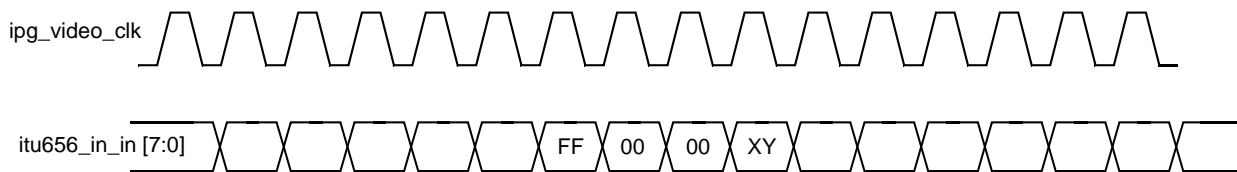


Figure 247. Location of Sync Preamble

Sync Preamble would come continuously for 4 clock cycles as shown in [Figure 247](#). Sync extraction is done using itu656_in. Sync Extraction identifies the horizontal and vertical blanking period using H and V field of the 'XYh' data as mentioned in Table 0-12

ITU-BT656 Sync preamble pattern (FFh 00h 00h) has to be masked out in the YCbCr data. The data stream must not include FFh 00h 00h as the valid pixel data to avoid malfunction.

Horizontal blanking period must be coming during the Vertical blanking period. Gap between 2 Horizontal blanking should be same during Vertical Blanking period as during line active.

During blanking period the sequence to sent is 80h 10h 80h 10h sequence. This sequence would be present both during line blanking and frame blanking period.

Framing Bit (F field in XYh) would be ignored during extraction. ECC error is detected. Any error in HSYNC or VSYNC bits in the stream are detected. Single bit errors are detected & corrected while two bit errors are only detected.

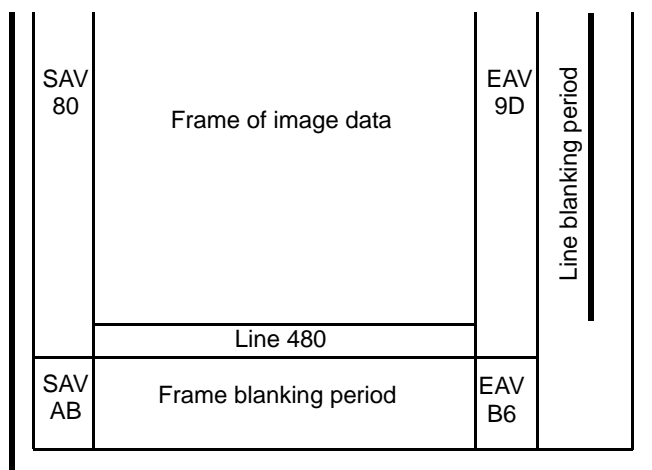


Figure 248. Relationship Between Hblank and Vblank in Internal Sync

21.4.1.4 Video In Format for External Sync Mode

In External sync Mode, the sensor inputs data in YCbCr422 format. The data can be in any of 8/10/12 bit format. This can be selected by Bit Width In field in status config register. Following suggests the setting of MJPEG Encoder for different bit widths

- 1) When data format is 8 bit, data is present in `itu656_in[11:4]` bits. The MJPEG Encoder must be configured to work in in Baseline(8 bit) mode.
- 2) When data format is 10 bits, the data is present `itu656_in[11:2]` bits. The design will append 2'b0 in the LSB's .The MJPEG encoder must be configured to work in Extended Sequential1 (12 bit) Mode
- 3)When data format is 12 bit, data is present in `itu656_in[11:0]` bits. The MJPEG Encoder must be configured to work in in Extended Sequential (12 bit) mode

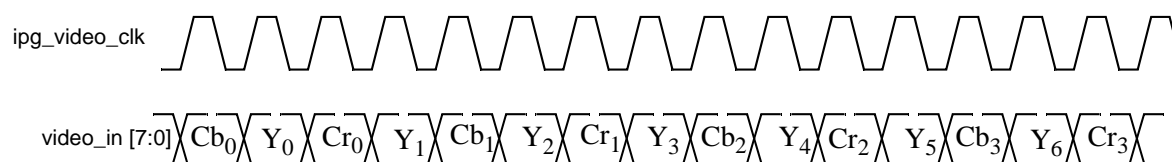


Figure 249. Data Format in External Sync YCbCr422 8 bit Mode

Figure 249 shows the data input from the camera interface in YcbCr422 format. This figure is also applicable for 10/12 bits mode.

NOTE

The data ordering for Chroma and Luma pixels is programmable. This is applicable for both internal and external sync modes. The ordering can be selected by a bit(`pixel_order`) in the status config register.

21.4.2 Circular buffer

The block encodes imager data, and stores in the internal circular buffer. The circular buffer is of size 2048x32 bits. The circular buffer is directly visible in the AHB address space of the device.

The circular buffer needs to be configured before starting the encoding process.

To start the circular buffer:

1. Write `dma_vstart_address`
2. Enable encoder to start on next frame, and set `dma_alarm_address` at the address when enough data is received to start data transfer to external.

Once the encoding starts, the processor accepts the `dma_alarm` interrupt. Whenever this interrupt is received, the processor does the following:

1. Rewrites the `dma_alarm_address`. This turns off the interrupt, and reenables it for later triggering. If the output buffer already passed the new alarm address, then the interrupt is requested again. In case of an address wrap around, the software needs to ensure proper alarm address handling.
2. Initiates a transfer, via DMA or ethernet, from the circular buffer to read the produced data. DMA or ethernet transfer increments addresses, as it is reading data from physical address of the RAM included in the video in block. Circular buffer implements a memory interface, no FIFO interface.

NOTE

While configuring the alarm address, the last alarm address before the wrap around must be 0x1FFC.

The block has the option to generate an interrupt after every picture, and stores the end-of-picture address in a register. So, the ethernet transmission can be synchronized to coincide with every encoded picture end. The block automatically pads every image frame length to be aligned to 32 bits. This padding is done by inserting FF bytes in front of the end-of-image marker. On reception of the end-of-picture interrupt, the `dma_vend_address` pointer will point to the first longword following the valid image.

The design supports 8/16/32 bits access on AHB bus. The MJPEG Encoder always outputs data in 16 bits. This data is converted to 32 bit before being written to the output buffer. It is therefore guaranteed that the write to the buffer will always be on the alternate clock cycle.

To provide minimum wait states on the read side, the design implements a prefetch buffer. Generally, the access to the output buffer is sequential. The prefetch buffer then fetches data corresponding to the next sequential addresses. However, it is possible to perform any non-sequential access to the AHB buffer with multiple wait states.

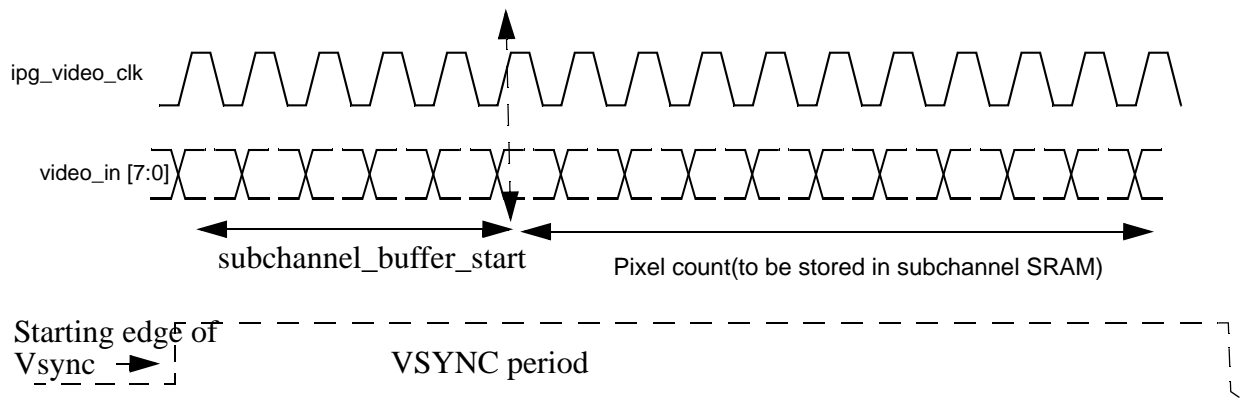
21.4.3 Subchannel Mode

Many imagers transmit data during the vertical blanking. The nature of the data can be imager register values or histogram data. As the video encoder does not do any interpretation of the data, it is referred to as the ‘subchannel’, without making any assumptions on the content of the data.

Subchannel data recovery is possible in the Data lines D[11:4]. The *subchan_data_req* flag needs to be written 1 every time the subchannel reception is requested.

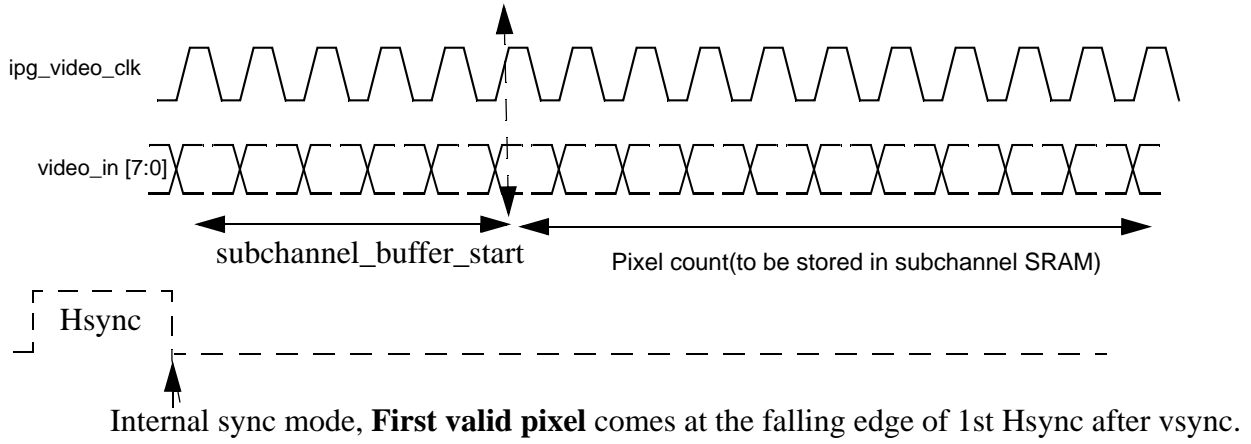
The encoder then follows the following steps:-

- 1) Check the *subchannel_start_point* field of status config register to know the starting point of pixel count as defined by the *subchannel_buffer_start* register.
- 2) If the *subchannel_start_point* bit is ‘0’, then it starts counting from the vsync’s starting edge irrespective of *sync_mode*. If the *subchannel_start_point* bit is ‘1’, then it starts counting from the first valid pixel of frame i.e. from first href’s starting edge after vsync in case *sync_mode* is ‘0’ {external sync} and first hsync’s negative edge after vsync in case *sync_mode* is ‘1’ {internal sync}, and .
- 3) Once the counting defined by *subchannel_buffer_start* is done, the pixel data of number of pixels defined by *pixel_count* register needs to be stored. For this, part *subchan_data_req* flag needs to be set to ‘1’. If the flag is set, then the data is written to the subchannel 64x32 SRAM.
- 4) After writing the complete data in the SRAM, *subchnl_irq* is generated which needs to be cleared by asserting the *subchnl_irq_clear* bit in status config register.
- 5) The *subchan_data_req* bit is automatically cleared on assertion of *subchnl_irq* and it needs to be set for every frame in which subchannel data is expected.



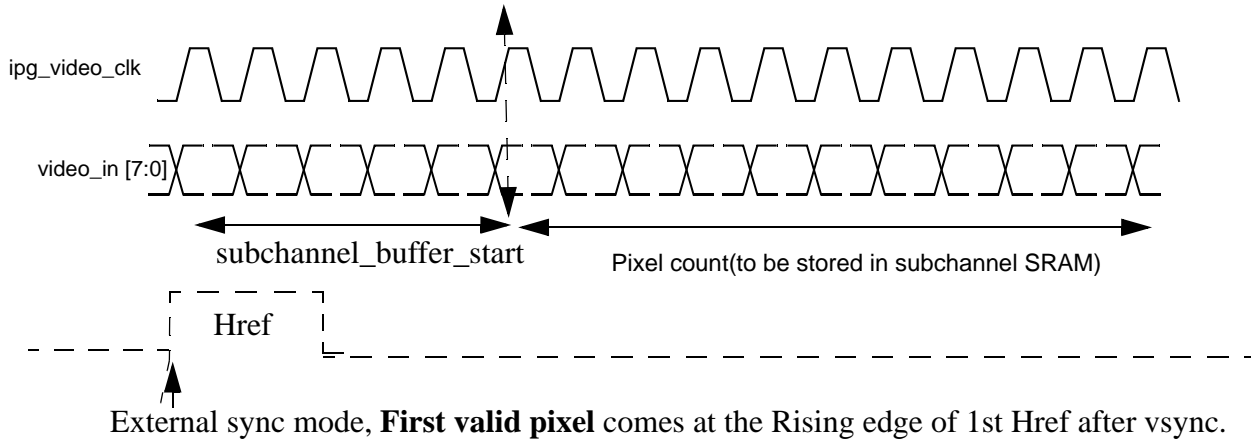
NOTE:- *Subchannel_start_point* bit = 1'b0

Figure 250. Subchannel data for External/Internal sync



NOTE:- Subchannel_start_point = 1

Figure 251. Internal Sync mode for Subchannel_start_point = 1



NOTE:- Subchannel_start_point = 1

Figure 252. External Sync mode for Subchannel_start_point = 1

21.4.4 Programming Sequence

The software should program the wrapper, memories and MJPEG registers in a specific sequence as mentioned below.

Initial configuration for the First Frame

1. Initialize Wrapper registers.(Buffer_restart=1 & Buffer_write_ON = 1). Make sure that Video Encoder ON bit is not SET.
2. Video Encoder ON = 1
3. Upload JPEGIn Configuration data to JPEGIN RAM.

NOTE

All contents of JPEGIn buffer (except last two words) are copied to Output buffer once the CAST and wrapper are enabled.

4. Set CONF = 1 (Autoclear = 1)
5. Set GO = 1 (AutoClear = 1)
6. Dma Alarm Address = xxxx

For follow up frames

1. EOI service routine
2. Read Bitrate of passed Frame.
3. Update Quantization Tables in JPEGIn of Wrapper.
4. Set CONF = 1 (Autoclear = 1)
5. Set GO = 1 (AutoClear = 1)
6. Set DMA Alarm Address = DMA Vend Address + xxxx
7. Return From Interrupt.

THIS PAGE INTENTIONALLY BLANK

Chapter 22

Integrated Interchip Sound (I²S) / Synchronous Audio Interface (SAI)

22.1 Introduction

The I²S (or I2S) module provides a Synchronous Audio Interface (SAI) that supports half-duplex serial interfaces with frame synchronization such as I2S, AC97, and CODEC/DSP interfaces.

22.1.1 Features

- Transmitter with independent Bit Clock and Frame Sync supporting 4 data channels
- Receiver with independent Bit Clock and Frame Sync supporting 4 data channels
- Maximum Frame Size of 16 Words
- Word size of between 8-bits and 32-bits Word size configured separately for first word and remaining words in frame
- Asynchronous 8 × 32-bit FIFO for each Transmit and Receive Channel
- Graceful restart after FIFO Error

22.1.2 Modes of Operation

The SAI operating modes include run mode and debug mode.

22.1.2.1 Run Mode

In run mode, the SAI Transmitter and Receiver operate normally.

22.1.2.2 Debug Mode

In Debug Mode, the SAI Transmitter and/or Receiver can continue operating provided the Debug Enable bit is set. When the Transmitter or Receiver Debug Enable bit is clear and Debug mode is entered, the SAI is disabled after completing the current Transmit or Receive Frame. The Transmitter and Receiver bit clocks are not affected by debug mode.

22.2 External signals

Name	Function	I/O	Reset	Pull
SAI_TX_BCLK	Transmit Bit Clock	I/O	0	—
SAI_TX_SYNC	Transmit Frame Sync	I/O	0	—
SAI_TX_DATA[3:0]	Transmit Data	O	0	—
SAI_RX_BCLK	Receive Bit Clock	I/O	0	—
SAI_RX_SYNC	Receive Frame Sync	I/O	0	—
SAI_RX_DATA[3:0]	Receive Data	I	0	—
SAI_MCLK	Audio Master Clock	I/O	0	—

22.3 Memory Map and Registers

Offset 0xFFFD_8000 (SAI 0) 0xFFFF0000 (SAI 1) 0xFFFF4000 (SAI 2)	Register	Access	Implemented in SAI0	Implemented in SAI 1/SAI 2
0x0000	SAI Transmit Control Register (I2S_TCSR)	R/W	Yes	Yes
0x0004	SAI Transmit Configuration 1 Register (I2S_TCR1)	R/W	Yes	Yes
0x0008	SAI Transmit Configuration 2 Register (I2S_TCR2)	R/W	Yes	Yes
0x000C	SAI Transmit Configuration 3 Register (I2S_TCR3)	R/W	Yes	Yes
0x0010	SAI Transmit Configuration 4 Register (I2S_TCR4)	R/W	Yes	Yes
0x0014	SAI Transmit Configuration 5 Register (I2S_TCR5)	R/W	Yes	Yes
0x0020	SAI Transmit Data Register (I2S_TDR0)	W (always reads zero)	Yes	Yes
0x0024	SAI Transmit Data Register (I2S_TDR1)	W (always reads zero)	Yes	No
0x0028	SAI Transmit Data Register (I2S_TDR2)	W (always reads zero)	Yes	No
0x002C	SAI Transmit Data Register (I2S_TDR3)	W (always reads zero)	Yes	No
0x0040	SAI Transmit FIFO Register (I2S_TFR0)	R	Yes	Yes

Offset 0xFFFD_8000 (SAI 0) 0xFFFF0000 (SAI 1) 0xFFFF4000 (SAI 2)	Register	Access	Implemented in SAI0	Implemented in SAI 1/SAI 2
0x0044	SAI Transmit FIFO Register (I2S_TFR1)	R	Yes	No
0x0048	SAI Transmit FIFO Register (I2S_TFR2)	R	Yes	No
0x004C	SAI Transmit FIFO Register (I2S_TFR3)	R	Yes	No
0x0060	SAI Transmit Mask Register (I2S_TMR)	R/W	Yes	Yes
0x0080	SAI Receive Control Register (I2S_RCSR)	R/W	Yes	Yes
0x0084	SAI Receive Configuration 1 Register (I2S_RCR1)	R/W	Yes	Yes
0x0088	SAI Receive Configuration 2 Register (I2S_RCR2)	R/W	Yes	Yes
0x008C	SAI Receive Configuration 3 Register (I2S_RCR3)	R/W	Yes	Yes
0x0090	SAI Receive Configuration 4 Register (I2S_RCR4)	R/W	Yes	Yes
0x0094	SAI Receive Configuration 5 Register (I2S_RCR5)	R/W	Yes	Yes
0x00A0	SAI Receive Data Register (I2S_RDR0)	R	Yes	Yes
0x00A4	SAI Receive Data Register (I2S_RDR1)	R	Yes	No
0x00A8	SAI Receive Data Register (I2S_RDR2)	R	Yes	No
0x00AC	SAI Receive Data Register (I2S_RDR3)	R	Yes	No
0x00C0	SAI Receive FIFO Register (I2S_RFR0)	R	Yes	Yes
0x00C4	SAI Receive FIFO Register (I2S_RFR1)	R	Yes	Yes
0x00C8	SAI Receive FIFO Register (I2S_RFR2)	R	Yes	Yes
0x00CC	SAI Receive FIFO Register (I2S_RFR3)	R	Yes	No
0x00E0	SAI Receive Mask Register (I2S_RMR)	R/W	Yes	Yes
0x0100	SAI MCLK Control Register (I2S_MCR)	R/W	Yes	Yes
0x0104	MCLK Divide Register (I2S_MDR)	R/W	Yes	Yes

22.3.1 SAI Transmit Control Register (I2S_TCSR)

22.3.2 SAI Transmit Configuration 1 Register (I2S_TCR1)

Offset: 0h

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		0	DBG E	BCE	0	0		SR	0			WSF	SEF	FEF	FWF	FRF
W							FR					w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									0				0			
W				WSI E	SEIE	FEIE	FWI E	FRIE							FWD E	FRD E
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 255. I2S_TCSR field descriptions

Field	Description
TE	Transmitter enable Enables/disables the transmitter. When software clears this bit, the transmitter remains enabled (and this bit remains set) until the end of the current frame. 0 — Transmitter is disabled. 1 — Transmitter is enabled, or transmitter has been disabled and not end of frame.
DBGE	Debug enable Enables/disables transmitter operation in debug mode. The transmit bit clock is not affected by debug mode. 0 — Transmitter is disabled in debug mode, after completing the current frame. 1 — Transmitter is enabled in debug mode.
BCE	Bit clock enable Enables the transmit bit clock, separately from the transmit enable. This bit is automatically set whenever the transmit enable is also set. When software clears this bit, the transmit bit clock remains enabled (and this bit remains set) until the end of the current frame. 0 — Transmit bit clock is disabled 1 — Transmit bit clock is enabled
FR	FIFO reset Resets the FIFO pointers. 0 — No effect. 1 — FIFO reset.
SR	Software reset When set, resets the internal transmitter logic including the FIFO pointers. Software visible-registers are not affected, except for the status registers. 0 — No effect. 1 — Software reset.
WSF	Word start flag Indicates that the start of the configured word has been detected. Write a logic one to this register bit to clear this flag. 0 — Start of word not detected. 1 — Start of word detected.

Table 255. I2S_TCSR field descriptions

Field	Description
SEF	Sync error flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic one to this register bit to clear this flag. 0 — Sync error not detected. 1 — Frame sync error detected.
FEF	FIFO error flag Indicates that an enabled transmit FIFO has underrun. Write a logic one to this register bit to clear this flag. 0 — Transmit underrun not detected. 1 — Transmit underrun detected.
FWF	FIFO warning flag Indicates that an enabled transmit FIFO is empty. 0 — No enabled transmit FIFO is empty. 1 — Enabled transmit FIFO is empty.
FRF	FIFO request flag Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0 — Transmit FIFO watermark not reached. 1 — Transmit FIFO watermark has been reached.
WSIE	Word start interrupt enable Enables/disables word start interrupts. 0 — Disables interrupt. 1 — Enables interrupt.
SEIE	Sync error interrupt enable Enables/disables sync error interrupts. 0 — Disables interrupt. 1 — Enables interrupt.
FEIE	FIFO error interrupt enable Enables/disables FIFO error interrupts. 0 — Disables the interrupt, 1 — Enables the interrupt.
FWIE	FIFO warning interrupt enable Enables/disables FIFO warning interrupts. 0 — Enables the interrupt. 1 — Disables the interrupt.
FRIE	FIFO request interrupt enable Enables/disables FIFO request interrupts. 0 — Disables the interrupt. 1 — Enables the interrupt.
FWDE	FIFO warning DMA enable Enables/disables DMA requests. 0 — Disables the DMA request. 1 — Enables the DMA request.
FRDE	FIFO request DMA enable Enables/disables DMA requests. 0 — Disables the DMA request. 1 — Enables the DMA request.

SAI Transmit Configuration 2 Register (I2S_TCR2)

Offset: 4h

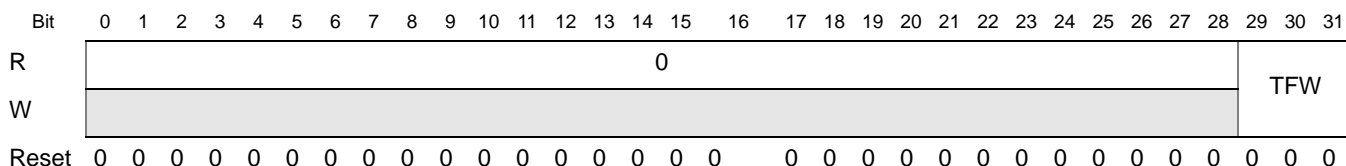


Table 256. I2S_TCR1 field descriptions

Field	Description
TFW	Transmit FIFO watermark Configures the watermark level for all enabled transmit channels.

SAI Transmit Configuration 3 Register (I2S_TCR3)

Offset: 8h

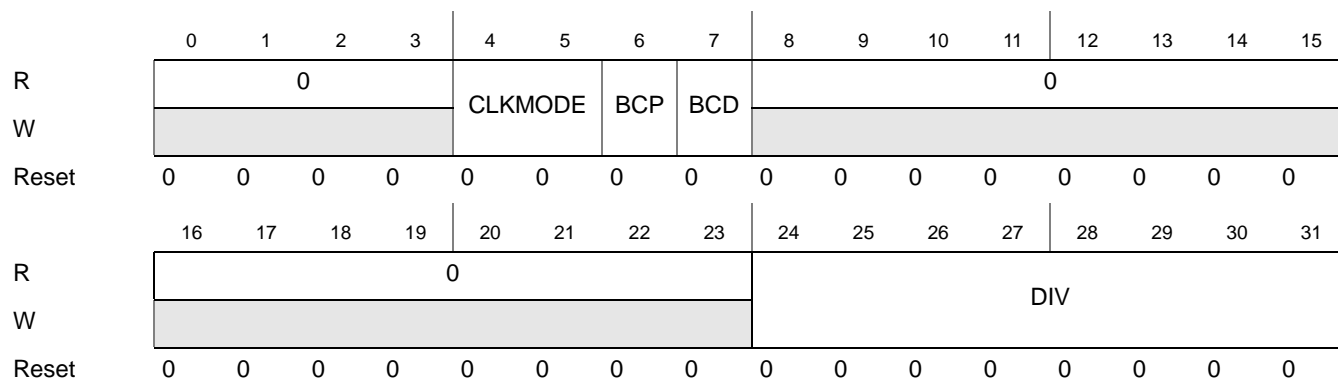


Table 257. I2S_TCR2 field descriptions

Field	Description
CLKMODE	Clocking mode When configured for external bit clock configures for asynchronous or synchronous operation. When configured for internal bit clock, selects the Audio Master Clock used to generate the internal bit clock. 00 — Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock). 01 — Synchronous with receiver (external bit clock) or Master Clock 1 selected (internal bit clock). 10 — Synchronous with another SAI transmitter (external bit clock) or Master Clock 2 selected (internal bit clock). 11 — Synchronous with another SAI receiver (external bit clock) or Master Clock 3 selected (internal bit clock).
BCP	Bit clock polarity Configures the polarity of the bit clock. 0 — Bit Clock is active high (drive outputs on rising edge and sample inputs on falling edge). 1 — Bit Clock is active low (drive outputs on falling edge and sample inputs on rising edge).

Table 257. I2S_TCR2 field descriptions

Field	Description
BCD	Bit clock direction Configures the direction of the bit clock. 0 — Bit clock is generated externally (slave mode). 1 — Bit clock is generated internally (master mode).
DIV	Bit clock divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is $(DIV + 1) * 2$.

NOTE: On SAI1/SAI2, the TCE field occupies only bit 16, and the WDFL field occupies only bit 0.

Offset: Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0											TCE				0											WDFL						
W	[Shaded]											TCE				[Shaded]											WDFL						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 258. I2S_TCR3 field descriptions

Field	Description
TCE	Transmit channel enable Enables a data channel for a transmit operation. A channel must be enabled before its FIFO can be accessed.
WDFL	Word flag configuration Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.

22.3.3 SAI Transmit Configuration 4 Register (I2S_TCR4)

NOTE: On SAI1/SAI2, the FRSZ field occupies only bit 16.

Offset: 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0											FRSZ				0			SYWD						0			M	F	0	F	F	
W	[Shaded]											FRSZ				[Shaded]			SYWD						[Shaded]			M	F	0	F	F	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 259. I2S_TCR4 field descriptions

Field	Description
FRSZ	Frame size Configures the number of words in each frame. The value written should be one less than the number of words in the frame (for example, write 0 for one word per frame). The maximum supported frame size is 16 words.
SYWD	Sync width Configures the length of the frame sync in number of bit clocks. The value written should be one less than the number of bit clocks (for example, write 0 for the frame sync to assert for one bit clock only). The sync width cannot be configured longer than the first word of the frame.
MF	MSB first Specifies whether the LSB or the MSB is transmitted/received first. 0 — LBS is transmitted/received first. 1 — MBS is transmitted/received first.
FSE	Frame sync early 0 — Frame sync asserts with the first bit of the frame. 1 — Frame sync asserts one bit before the first bit of the frame.
FSP	Frame sync polarity Configures the polarity of the frame sync. 0 — Frame sync is active high. 1 — Frame sync is active low.
FSD	Frame sync direction Configures the direction of the frame sync. 0 — Frame Sync is generated externally (slave mode). 1 — Frame Sync is generated internally (master mode).

22.3.4 SAI Transmit Configuration 5 Register (I2S_TCR5)

Offset: 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			WNW				0			WOW				0			FBT				0										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 260. I2S_TCR5 field descriptions

Field	Description
WNW	Word N width Configures the number of bits in each word, for each word except the first in the frame. The value written should be one less than the number of bits per word. This field must be configured greater than or equal to Word 0 Width even when there is only one word in each frame. Words of fewer than 8 bits wide are not supported.
W0W	Word 0 width Configures the number of bits in the first word in each frame. The value written should be one less than the number of bits in the first word. Words of less than 8 bits wide are not supported if there is only one word per frame.
FBT	First bit shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written should be greater than or equal to the word width when configured for MSB First. The value written should be less than or equal to (31 - word with) when configured for LSB First.

22.3.5 SAI Transmit Data Register (I2S_TDR)

Offset: 20h (TDR0 on SAI0 or TDR on SAI1/SAI2)
 24h (TDR1 on SAI0)
 28h (TDR2 on SAI0)
 2Ch (TDR3 on SAI0)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0																																	
W	TDR[31:0]																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 261. I2S_TDR_n field descriptions

Field	Description
TDR[31:0]	Transmit data register Writes to this register when the transmit data channel is enabled and not full will push the data written into the transmit FIFO. Otherwise, writes to this register are ignored.

22.3.6 SAI Transmit FIFO Register (I2S_TFR)

The MSB of the read pointer and write pointer is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Offset: 40h (TFR0 on SAI0 or TFR on SAI1/SAI2)
 44h (TFR1 on SAI0)
 48h (TFR2 on SAI0)
 4Ch (TFR3 on SAI0)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0											WFP				0								RFP									
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 262. I2S_TFRn field descriptions

Field	Description
WFP	Write FIFO pointer FIFO write pointer for transmit data channel.
RFP	Read FIFO pointer FIFO read pointer for transmit data channel.

22.3.7 SAI Transmit Mask Register (I2S_TMR)

NOTE: On SAI1/SAI2, the TWMM field occupies only bits 1-0.

Offset: 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TWMM																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 263. I2S_TMR field descriptions

Field	Description
TWMM	Transmit word mask For each word in the frame, configures whether the transmit word is masked. 0 — Word N is enabled. 1 — Word N is masked. The transmit data pins are tri-stated when masked.

22.3.8 SAI Receive Control Register (I2S_RCSR)

Offset: 80h

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

R	RE	0	DBGE	BCE	0	0	FR	SR	0	WSF	SEF	FEF	FWF	FRF		
W										w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			WSIE	SEIE	FEIE	FWIE	FRIE	0			0		FWE	FRD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 264. I2S_RCSR field descriptions

Field	Description
RE	Receiver enable Enables/disables the receiver. When software clears this bit, the receiver remains enabled (and this bit remains set) until the end of the current frame. 0 — Receiver is disabled. 1 — Receiver is enabled, or receiver has been disabled and not end of frame.
DBGE	Debug enable Enables/disables receiver operation in debug mode. The receive bit clock is not affected by debug mode. 0 — Receiver is disabled in debug mode, after completing the current frame. 1 — Receiver is enabled in debug mode.
BCE	Bit clock enable Enables the receive bit clock, separately from the receive enable. This bit is automatically set whenever the receive enable is also set. When software clears this bit, the receive bit clock remains enabled (and this bit remains set) until the end of the current frame. 0 — Receive bit clock is disabled 1 — Receive bit clock is enabled
FR	FIFO reset Resets the FIFO pointers. 0 — No effect. 1 — FIFO reset.
SR	Software reset When set, resets the internal receiver logic including the FIFO pointers. Software visible-registers are not affected, except for the status registers. 0 — No effect. 1 — Software reset.
WSF	Word start flag Indicates that the start of the configured word has been detected. Write a logic one to this register bit to clear this flag. 0 — Start of word not detected. 1 — Start of word detected.
SEF	Sync error flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic one to this register bit to clear this flag. 0 — Sync error not detected. 1 — Frame sync error detected.

Table 264. I2S_RCSR field descriptions

Field	Description
FEF	FIFO error flag Indicates that an enabled receive FIFO has overflowed. Write a logic one to this register bit to clear this flag. 0 — Receive underrun not detected. 1 — Receive underrun detected.
FWF	FIFO warning flag Indicates that an enabled receive FIFO is full. 0 — No enabled receive FIFO is full. 1 — Enabled receive FIFO is full.
FRF	FIFO request flag Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark. 0 — Receive FIFO watermark not reached. 1 — Receive FIFO watermark has been reached.
WSIE	Word start interrupt enable Enables/disables word start interrupts. 0 — Disables interrupt. 1 — Enables interrupt.
SEIE	Sync error interrupt enable Enables/disables sync error interrupts. 0 — Disables interrupt. 1 — Enables interrupt.
FEIE	FIFO error interrupt enable Enables/disables FIFO error interrupts. 0 — Disables the interrupt, 1 — Enables the interrupt.
FWIE	FIFO warning interrupt enable Enables/disables FIFO warning interrupts. 0 — Enables the interrupt. 1 — Disables the interrupt.
FRIE	FIFO request interrupt enable Enables/disables FIFO request interrupts. 0 — Disables the interrupt. 1 — Enables the interrupt.
FWDE	FIFO warning DMA enable Enables/disables DMA requests. 0 — Disables the DMA request. 1 — Enables the DMA request.
FRDE	FIFO request DMA enable Enables/disables DMA requests. 0 — Disables the DMA request. 1 — Enables the DMA request.

22.3.9 SAI Receive Configuration 1 Register (I2S_RCR1)

Offset: 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												RFW			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22.3.10 SAI Receive Configuration 2 Register (I2S_RCR2)

Table 265. I2S_RCR1 field descriptions

Field	Description
RFW	Receive FIFO watermark Configures the watermark level for all enabled receiver channels.

SAI Receive Configuration 3 Register (I2S_RCR3)

Offset: 88h

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				CLKMODE	BCP	BCD	0								
W					CLKMODE	BCP	BCD									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 266. I2S_RCR2 field descriptions

Field	Description
CLKMODE	Clocking mode When configured for external bit clock configures for asynchronous or synchronous operation. When configured for internal bit clock, selects the Audio Master Clock used to generate the internal bit clock. 00 — Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock). 01 — Synchronous with transmitter (external bit clock) or Master Clock 1 selected (internal bit clock). 10 — Synchronous with another SAI receiver (external bit clock) or Master Clock 2 selected (internal bit clock). 11 — Synchronous with another SAI transmitter (external bit clock) or Master Clock 3 selected (internal bit clock).
BCP	Bit clock polarity Configures the polarity of the bit clock. 0 — Bit Clock is active high (drive outputs on rising edge and sample inputs on falling edge). 1 — Bit Clock is active low (drive outputs on falling edge and sample inputs on rising edge).

Table 266. I2S_RCR2 field descriptions

Field	Description
BCD	Bit clock direction Configures the direction of the bit clock. 0 — Bit clock is generated externally (slave mode). 1 — Bit clock is generated internally (master mode).
DIV	Bit clock divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

NOTE: On SAI1/SAI2, the RCE field occupies only bit 16, and the WDFL field occupies only bit 0.

SAI Receive Configuration 4 Register (I2S_RCR4)

Offset: 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RCE	0												WDFL						
W	[Shaded]													[Shaded]																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 267. I2S_RCR3 field descriptions

Field	Description
RCE	Receive channel enable Enables a data channel for a receive operation. A channel must be enabled before its FIFO can be accessed.
WDFL	Word flag configuration Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.

NOTE: On SAI1/SAI2, the FRSZ field occupies only bit 16.

SAI Receive Configuration 5 Register (I2S_RCR5)

Offset: 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FRSZ	0			SYWD	0			M	F	0	F	F							
W	[Shaded]													[Shaded]						F	S	[Shaded]	F	S								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 268. I2S_RCR4 field descriptions

Field	Description
FRSZ	Frame size Configures the number of words in each frame. The value written should be one less than the number of words in the frame (for example, write 0 for one word per frame). The maximum supported frame size is 16 words.
SYWD	Sync width Configures the length of the frame sync in number of bit clocks. The value written should be one less than the number of bit clocks (for example, write 0 for the frame sync to assert for one bit clock only). The sync width cannot be configured longer than the first word of the frame.
MF	MSB first Specifies whether the LSB or the MSB is transmitted/received first. 0 — LBS is transmitted/received first. 1 — MBS is transmitted/received first.
FSE	Frame sync early 0 — Frame sync asserts with the first bit of the frame. 1 — Frame sync asserts one bit before the first bit of the frame.
FSP	Frame sync polarity Configures the polarity of the frame sync. 0 — Frame sync is active high. 1 — Frame sync is active low.
FSD	Frame sync direction Configures the direction of the frame sync. 0 — Frame Sync is generated externally (slave mode). 1 — Frame Sync is generated internally (master mode).

SAI Receive Data Register (I2S_RDR)

Offset: 94h

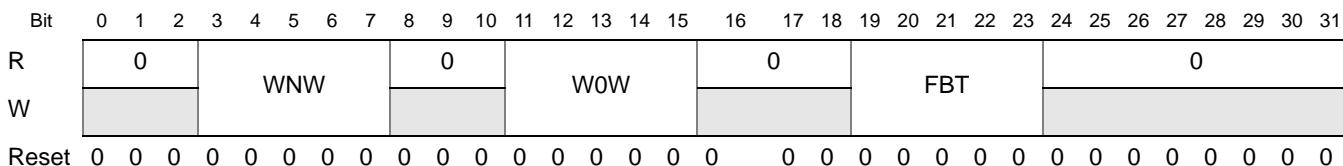


Table 269. I2S_RCR5 field descriptions

Field	Description
WNW	Word N width Configures the number of bits in each word, for each word except the first in the frame. The value written should be one less than the number of bits per word. This field must be configured greater than or equal to Word 0 Width even when there is only one word in each frame. Words of fewer than 8 bits wide are not supported.

Table 269. I2S_RCR5 field descriptions

Field	Description
WOW	Word 0 width Configures the number of bits in the first word in each frame. The value written should be one less than the number of bits in the first word. Words of less than 8 bits wide are not supported if there is only one word per frame.
FBT	First bit shifted Configures the bit index for the first bit received for each word in the frame. If configured for MSB First. The index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written should be greater than or equal to the word width when configured for MSB First. The value written should be less than or equal to (31 - word with) when configured for LSB First.

SAI Receive FIFO Register (I2S_RFR)

Offset: A0h (RDR0 on SAI0 or RDR on SAI1/SAI2)
 A4h (RDR1 on SAI0)
 A8h (RDR2 on SAI0)
 ACh (RDR3 on SAI0)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	RDR[31:0]																																
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 270. I2S_RDRn field descriptions

Field	Description
RDR[31:0]	Receive data register Reads from this register when the receive data channel is enabled and not empty return the data from the top of the receive FIFO. Otherwise, writes to this register are ignored.

The MSB of the read pointer and write pointer is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

SAI Receive Mask Register (I2S_RMR)

Offset: C0h (RFR0 on SAI0 or RFR on SAI1/SAI2)
 C4h (RFR1 on SAI0)
 C8h (RFR2 on SAI0)
 CCh (RFR3 on SAI0)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					WFP					0					RFP																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 271. I2S_RFR_n field descriptions

Field	Description
WFP	Write FIFO pointer FIFO write pointer for receive data channel.
RFP	Read FIFO pointer FIFO read pointer for receive data channel.

NOTE: On SAI1/SAI2, the RWM field occupies only bits 1-0.

SAI MCLK Control Register (I2S_MCR)

Offset: E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															RWM																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 272. I2S_RMR field descriptions

Field	Description
RWM	Receive word mask For each word in the frame, configures if the receive word is masked. 0 — Word N is enabled. 1 — Word N is masked.

MCLK Divide Register (I2S_MDR)

Offset: 100h

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

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

Table 273. I2S_MCR field descriptions

Field	Description
DUF	Divider Update Flag Provides the status of on-the-fly updates to the MCLK Divider ratio. 0 — MCLK Divider ratio is not being updated currently. 1 — MCLK Divider ratio is updating on-the-fly. Further updates to the MCLK Divider ratio are blocked while this flag remains set.
MOE	MCLK Output Enable Enables the MCLK Divider and configures the SAI_MCLK pin as an output. When software clears this bit, this bit remains set until the MCLK divider is fully disabled. 0 — SAI_MCLK pin is configured as an input that bypasses the MCLK Divider. 1 — SAI_MCLK pin is configured as an output from the MCLK Divider and the MCLK Divider is enabled.
MICS	MCLK Input Clock Select Selects the clock input to the MCLK Divider. This field cannot be changed when the MCLK divider is enabled. 00 — MCLK Divider input clock 0 selected. 01 — MCLK Divider input clock 1 selected. 10 — MCLK Divider input clock 2 selected. 11 — MCLK Divider input clock 3 selected.

Functional description

Offset: 104h

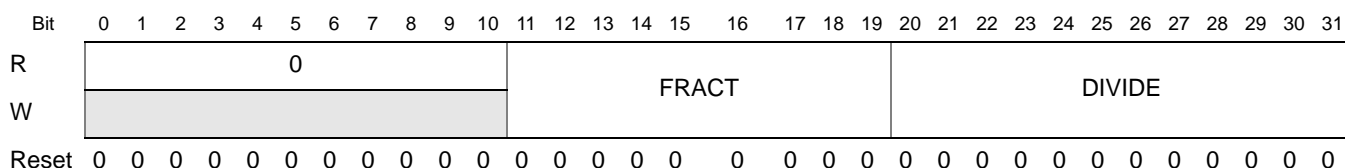


Table 274. I2S_MDR field descriptions

Field	Description
FRACT	MCLK Fraction The MCLK FRACT must be set equal or less than the MCLK DIVIDE. Sets the MCLK divide ratio such that: $MCLK\ output = MCLK\ input * ((FRACT + 1) / (DIVIDE + 1))$
DIVIDE	MCLK Divide The MCLK FRACT must be set equal or less than the MCLK DIVIDE. Sets the MCLK divide ratio such that: $MCLK\ output = MCLK\ input * ((FRACT + 1) / (DIVIDE + 1))$

22.3.11 SAI clocking

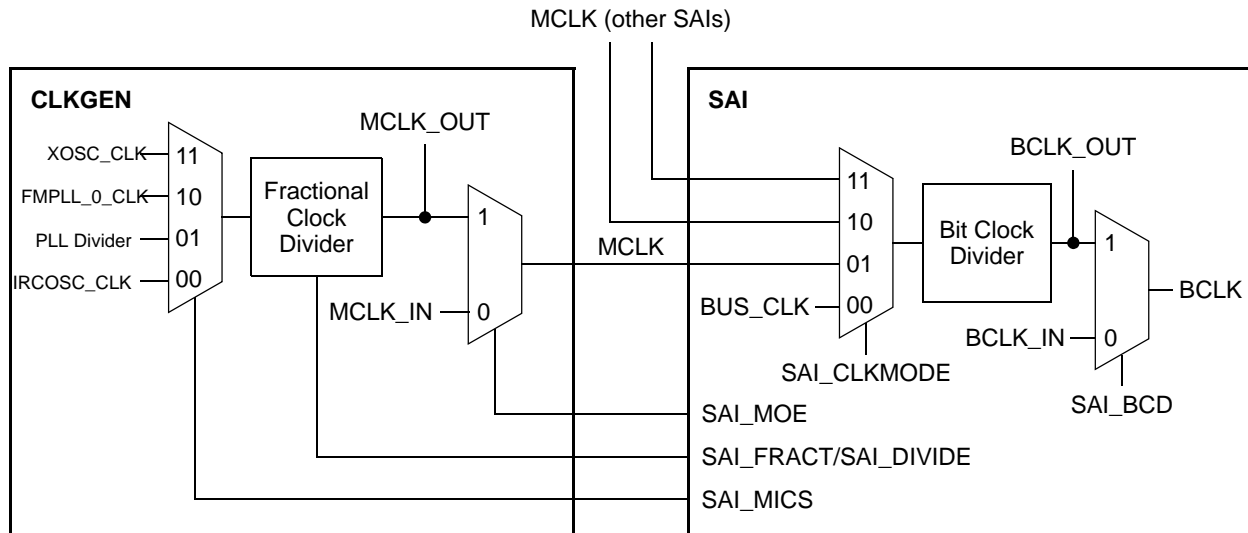
The SAI clocks include the audio master clock, the bit clock, and the bus clock.

22.3.11.1 Audio Master Clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally-generated bit clock. The master clock used for the transmitter and receiver can be selected independently from between the bus clock or up to three audio master clocks shared between different SAI instantiations. The audio master clock can also be an output to a pin or an input from a pin.

Each SAI peripheral can control the input clock selection, pin direction and divide ratio of one audio master clock. The input clock selection and pin direction cannot be altered if an SAI using that audio master clock has been enabled. The master clock divide ratio can be altered while an SAI is using that master clock.

Refer to [Chapter 4, "Clock Architecture"](#), for information. A typical implementation appears in the following figure.



22.3.11.2 Bit Clock

The SAI transmitter and receiver support asynchronous free running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter or between multiple SAI peripherals.

Externally generated bit clocks should be enabled before the SAI transmitter or receiver is enabled and disabled after the SAI transmitter or receiver is disabled and has completed the current frame as indicated by the transmitter or receiver enable bit.

22.3.11.3 Bus Clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

22.3.12 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

22.3.12.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

22.3.12.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after the Transmit FIFO Error Flag is set, and before the FIFO is re-initialized and the Error Flag is cleared. The FIFO Reset is asserted for one cycle only.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the Receive FIFO Error Flag is set and any remaining data has been read from the FIFO, and before the Error Flag is cleared. The FIFO Reset is asserted for one cycle only.

22.3.13 Synchronous Modes

The SAI transmitter and receiver can operate synchronously to each other or synchronously to other SAI peripherals.

22.3.13.1 Synchronous Mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync and controlled by the same module enable.

If the transmitter bit clock and frame sync are to be used by both then the transmitter should be configured for asynchronous operation and the receiver for synchronous operation. In synchronous mode, the receiver is only enabled when both the transmitter and receiver are both enabled. It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both then the receiver should be configured for asynchronous operation and the transmitter for synchronous operation. In synchronous mode, the transmitter is only enabled when both the receiver and transmitter are both enabled. It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode only the bit clock, frame sync and module enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers should be configured consistently across both the transmitter and receiver.

22.3.13.2 Multiple SAI Synchronous Mode

Synchronous operation between multiple SAI peripherals is not supported on all devices, and requires the source of the bit clock and frame sync to be configured for asynchronous operation and the remaining users of the bit clock and frame sync to be configured for synchronous operation.

Synchronous operation between multiple SAI transmitters or receivers also requires the source of the bit clock and frame sync to be enabled for any of the synchronous transmitters or receivers to also be enabled. It is recommended that the source of the bit clock and frame sync is the last enabled and the first disabled.

When operating in synchronous mode only the bit clock, frame sync and module enable are shared. The separate SAI peripherals otherwise operate independently, although configuration registers should be configured consistently across both the transmitter and receiver.

22.3.14 Frame sync configuration

The Frame Sync signal is used to indicate the start of each Frame. A valid Frame Sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the Transmitter or Receiver cannot be busy with a previous frame. A valid Frame Sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the Transmitter or Receiver.

The Transmitter and Receiver Frame Sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Asserts with first bit in frame or asserts one bit early
- Asserts for between 1 bit clock and first word length
- Frame length can be configured from 1 word per frame to 16 words per frame
- Word length can be configured to support from 8 bits to 32 bits per word
 - First word length and remaining word lengths can be configured separately
- Can be configured for Most Significant Bit first or Least Significant Bit first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

22.3.15 Data FIFO

22.3.15.1 Data alignment

Each transmit and receive channel includes a FIFO of size 8×32 -bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers. Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration is illustrated in [Figure 253](#) for MSB First configurations.

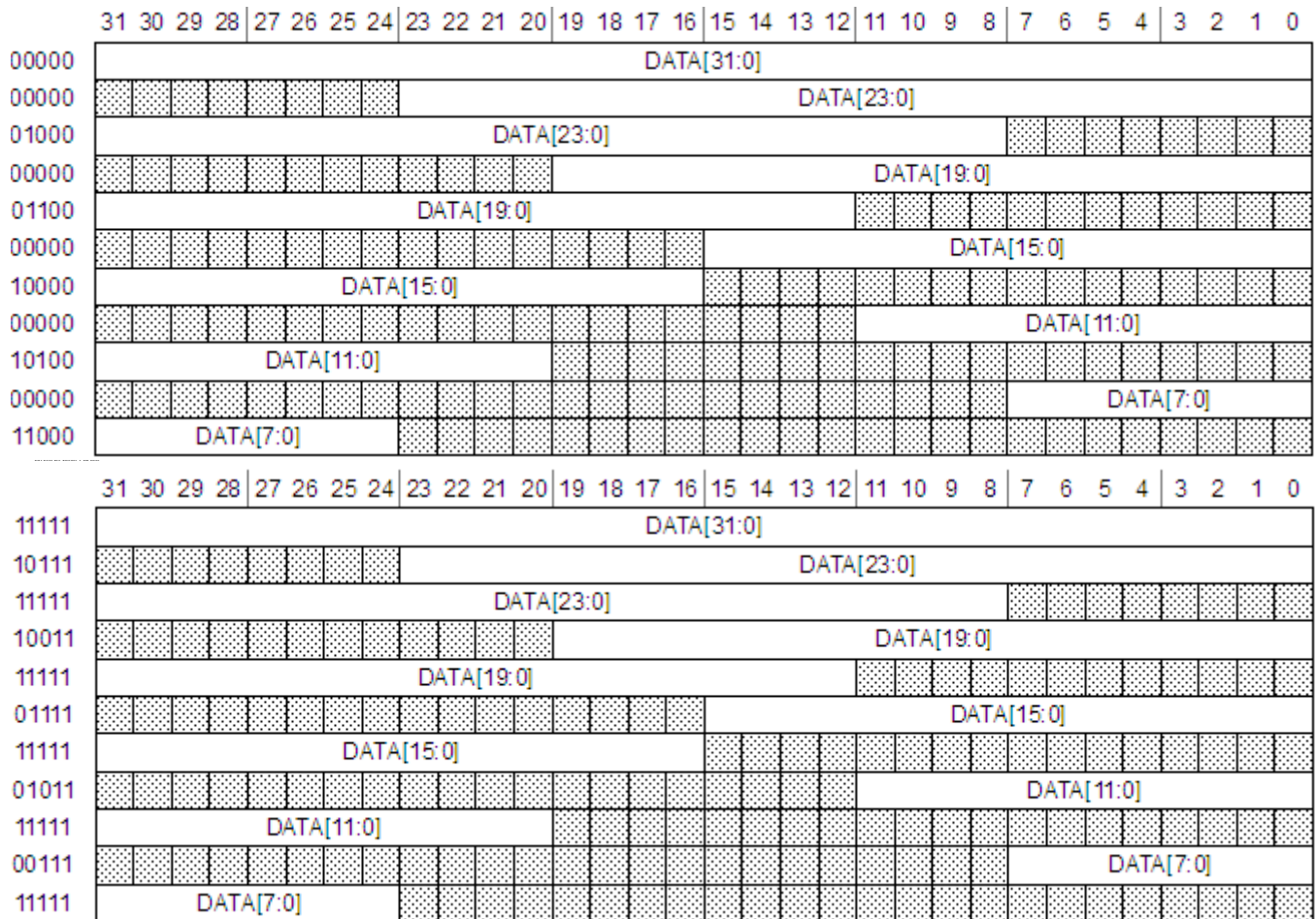


Figure 253. SAI First Bit Shifted, MSB First

22.3.15.2 FIFO pointers

When writing to the Transmit Data Register (TDR), the write FIFO pointer increments after each valid write. The SAI supports 8-bit and 16-bit writes to TDR for transmitting 8-bit and 16-bit data respectively.

Writes to the Transmit Data Register are ignored if the corresponding Transmit Channel Enable is clear or if the FIFO is full. If the Transmit FIFO is empty, the Transmit Data Register must be written at least three bit clocks before the start of the next unmasked word to avoid a FIFO underrun.

When reading the Receive Data Register (RDR), the read FIFO pointer increments after each valid read. The SAI supports 8-bit and 16-bit reads from RDR for receiving 8-bit and 16-bit data respectively.

Reads from the Receive Data Register are ignored if the corresponding Receive Channel Enable is clear or if the FIFO is empty. If the Receive FIFO is full, the Receive Data Register must be read at least three bit clocks before the end of an unmasked word to avoid a FIFO overrun.

22.3.16 Word mask register

The SAI transmitter and receiver each contain a word mask register that can be used to mask any word in the frame. Since the Word Mask Register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The transmitter word mask causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The receiver word mask causes the received data for each selected word to be discarded and not written to the receive FIFO.

22.3.17 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags.

22.3.17.1 FIFO data ready flag

The FIFO data ready flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit data ready flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive data ready flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO data ready flag can generate an interrupt or a DMA request.

22.3.17.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

22.3.17.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels repeat the last valid word read from the transmit FIFO until the transmit FIFO error flag is cleared and the start of the next transmit frame. All enabled transmit FIFOs should be reset and initialized with new data before the transmit FIFO error flag is cleared.

The receive FIFO error flag is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until the receive FIFO error flag is cleared and the start of the next receive frame. All enabled receive FIFOs should be emptied before the receive FIFO error flag is cleared.

The FIFO error flag can generate an interrupt only.

22.3.17.4 Sync error flag

The sync error flag is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. The transmitter or receiver continues checking for frame sync assertion at the end of each frame (or when idle) when the sync error flag is set.

The sync error flag can generate an interrupt only.

22.3.17.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

Chapter 23

SAI Instantiation

23.1 Introduction

This section summarizes how the module has been configured in MPC5604E. For a comprehensive description of the module itself, see the module's dedicated chapter.

Table 275. Reference links to related information

Topic	Related Module	Reference
Module Description	Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI)	Chapter 22, "Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI)"
System Memory Map	Peripheral memory map	Chapter 2, "Memory Map"
Clocking		Chapter 4, "Clock Architecture"
Signal Multiplexing	Pin muxing	Chapter 3, "Signal Description"

23.1.1 SAI/I2S Overview

MPC5604E instantiates three Synchronous Audio Interface (SAI) modules. The first Synchronous Audio Interface (SAI0) module supports up to 4 channels either with parallel data bits or using the AC97 mode with a single data bit and time division multiplexing (TDM).

In addition two more Synchronous Audio Interfaces are instantiated (SAI1 and SAI2) allowing one channel input or output each. These two modules can operate in a synchronous or asynchronous mode. In synchronous mode, SAI2 can control either SAI1 and SAI0 or only SAI1.

In total, MPC5604E can support up to 6 (stereo) audio channels (up to 6 data pins).

[Table 276](#) compares the SAI instances in the device.

Table 276. Comparison between SAI0 and SAI1/SAI2

Features	SAI0	SAI1	SAI2
Channel supported	4	1	1
Frame Synchronization	I2S, AC97, and CODEC/DSP interfaces	I2S and CODEC/DSP interfaces	I2S and CODEC/DSP interfaces
Frame Length	Maximum is 16 Words per frame	Maximum is 2 Words per frame	Maximum is 2 Words per frame

Table 276. Comparison between SAI0 and SAI1/SAI2

Features	SAI0	SAI1	SAI2
Synchronous Mode	SAI0 supports this mode as it has four dedicated pins.	Does not support this mode.	Does not support this mode.
Multiple SAI Synchronous Mode	—	—	Only SAI 2 can be the master for the multiple SAI synchronous modes.

23.1.2 External Signals Multiplexing

- SAI_TX_BCLK is muxed to SAI_RX_BCLK
- SAI_TX_SYNC is muxed to SAI_RX_SYNC
- SAI_TX_DATA[3:0] is muxed to SAI_RX_DATA[3:0]

23.1.3 SAI/I2S Clocking

In addition to the bus clock, the I2S/SAI module has different clock sources for master clock generation. The I2S/SAI clock is generated either from the external (I2S clock) or from the internal fractional clock divider (FCD).

23.1.3.1 SAI/I2S Clock Selection

MCLK Input Clock Select bit of MCLK Control Register (MCR[MICS]) bit selects the clock input to the FCD.

Table 277 shows the MCR[MICS] settings for MPC5604E.

Table 277. MCLK Input Clock Select

Value	Description
00	IRCOSC_CLK (16 MHz) clock is selected
01	PLL divider clock is selected
10	FMPLL_0_CLK clock is selected
11	XOSC_CLK clock is selected

23.1.3.2 CLKMODE in SAI/I2S TCR2 Register

CLKMODE bit of the Transmit Configuration 2 Register (TCR2[CLKMODE]) when configured for external bit clock configures asynchronous or synchronous operation. When configured for internal bit clock, selects the Audio Master Clock used to generate the internal bit clock.

Table 278 shows the TCR2[CLKMODE] settings for SAI0.

Table 278. CLKMODE configuration for SAI0

Value	Description
00	Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock).
01	Synchronous with receiver (external bit clock) or MCLK for SAI0 (Refer to PCR[36] of Pin Muxing Table in Chapter 3, "Signal Description")
10	Synchronous with the SAI2 transmitter (external bit clock) or MCLK of SAI1 (Refer to PCR[15] of Pin Muxing Table in Chapter 3, "Signal Description")
11	Synchronous with SAI2 receiver (external bit clock) or MCLK of SAI2 (Refer to PCR[10] of Pin Muxing Table in Chapter 3, "Signal Description")

Table 279 shows the TCR2[CLKMODE] settings for SAI1.

Table 279. CLKMODE configuration for SAI1

Value	Description
00	Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock).
01	Synchronous with receiver (external bit clock) or MCLK for SAI1 (Refer to PCR[15] of Pin Muxing Table in Chapter 3, "Signal Description")
10	Synchronous with the SAI2 transmitter (external bit clock) or MCLK of SAI0 (Refer to PCR[36] of Pin Muxing Table in Chapter 3, "Signal Description")
11	Synchronous with SAI2 receiver (external bit clock) or MCLK of SAI2 (Refer to PCR[10] of Pin Muxing Table in Chapter 3, "Signal Description")

Table 280 shows the TCR2[CLKMODE] settings for SAI2.

Table 280. CLKMODE configuration for SAI2

Value	Description
00	Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock).
01	Synchronous with receiver (external bit clock) or MCLK for SAI2 (Refer to PCR[36] of Pin Muxing Table in Chapter 3, "Signal Description")
10	Reserved (external bit clock) or MCLK of SAI0
11	Reserved (external bit clock) or MCLK of SAI1

23.1.3.3 CLKMODE in SAI/I2S RCR2 Register

Table 281 shows RCR2[CLKMODE] settings in MPC5604E.

Table 281. RCR2[CLKMODE] configuration for SAI0

Value	Description
00	Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock).
01	Synchronous with transmitter (external bit clock) or MCLK for SAI0 (Refer to PCR[36] of Pin Muxing Table in Chapter 3, "Signal Description")

Table 281. RCR2[CLKMODE] configuration for SAI0

Value	Description
10	Synchronous with the receiver of SAI2 or MCLK of SAI1 (Refer to PCR[15] of Pin Muxing Table in Chapter 3, "Signal Description")
11	Synchronous with the transmitter of SAI2 or MCLK of SAI2 (Refer to PCR[10] of Pin Muxing Table in Chapter 3, "Signal Description")

Table 282 shows the RCR2[CLKMODE] settings for SAI1.

Table 282. RCR2[CLKMODE] configuration for SAI1

Value	Description
00	Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock).
01	Synchronous with transmitter (external bit clock) or or MCLK for SAI1 (Refer to PCR[15] of Pin Muxing Table in Chapter 3, "Signal Description")
10	Synchronous with the receiver of SAI2 or MCLK of SAI0 (Refer to PCR[36] of Pin Muxing Table in Chapter 3, "Signal Description")
11	Synchronous with the transmitter of SAI2 or MCLK of SAI2 (Refer to PCR[10] of Pin Muxing Table in Chapter 3, "Signal Description")

Table 283 shows the RCR2[CLKMODE] settings for SAI2.

Table 283. RCR2[CLKMODE] configuration for SAI2

Value	Description
00	Asynchronous mode (external bit clock) or Bus Clock selected (internal bit clock).
01	Synchronous with receiver (external bit clock) or MCLK for SAI2 (Refer to PCR[36] of Pin Muxing Table in Chapter 3, "Signal Description")
10	Reserved (external bit clock) or MCLK of SAI0
11	Reserved (external bit clock) or MCLK of SAI1

23.1.3.4 Configuring clock source for SAI/I2S audio master clock

To configure clock source for SAI/I2S audio master clock:

1. Configure MCR:MICS of the [SAI MCLK Control Register \(I2S_MCR\)](#) to select clock source of the FCD.
2. Configure the [MCLK Divide Register \(I2S_MDR\)](#) to set Divide and Fraction.
3. Configure MCR: MOE of the [SAI MCLK Control Register \(I2S_MCR\)](#) to enable clock.

Chapter 24

Deserial Serial Peripheral Interface (DSPI)

24.1 Introduction

This chapter describes the deserial serial peripheral interface (DSPI), which provides a synchronous serial bus for communication between the MCU and an external peripheral device.

24.2 Block diagram

A block diagram of the DSPI is shown in [Figure 254](#).

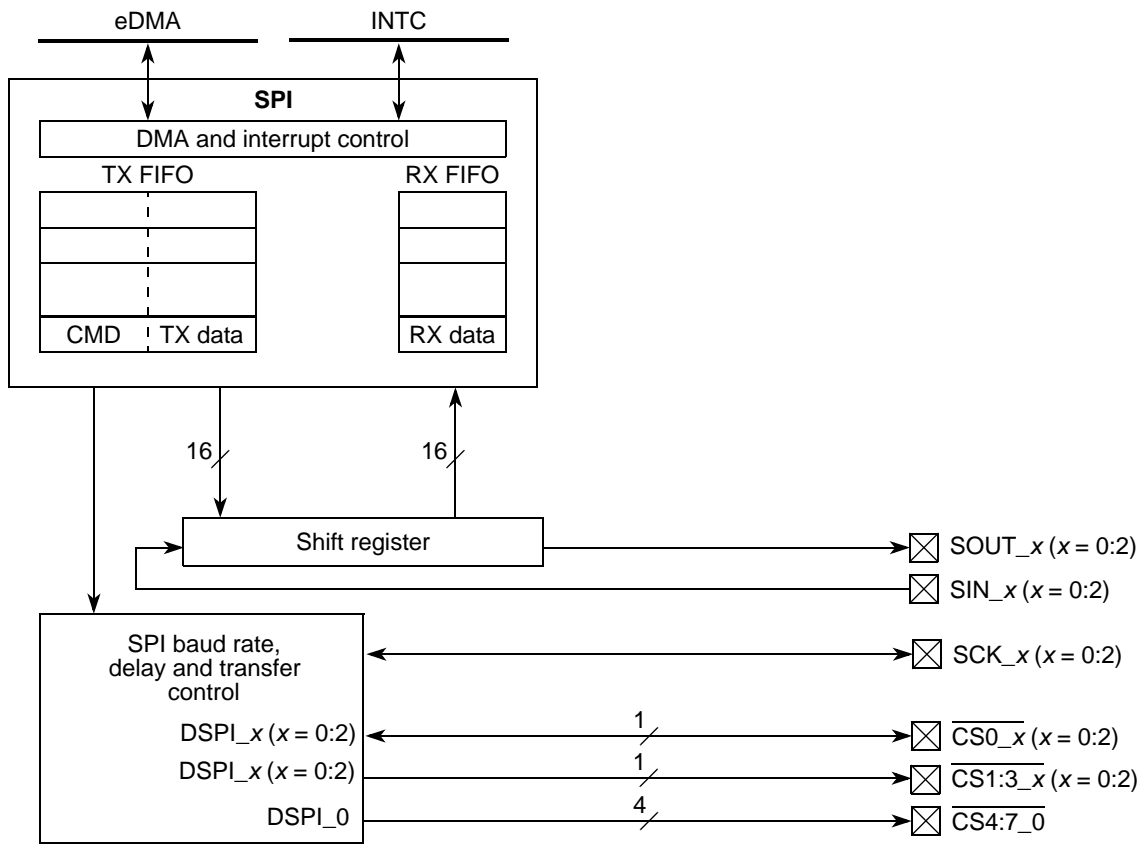


Figure 254. DSPI block diagram

24.3 Overview

The register content is transmitted using an SPI protocol. There are three DSPI modules (DSPI_0, DSPI_1, and DSPI_2) on the device. The modules are identical except that DSPI_0 has four additional chip select (\overline{CS}) lines.

The DSPI module in MPC5604E supports the SPI configuration. In SPI configuration, the DSPI operates as an SPI with queue support.

For queued operations, the SPI queues reside in internal SRAM that is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished through the use of the eDMA controller or through host software.

Figure 255 shows a DSPI with external queues in internal SRAM.

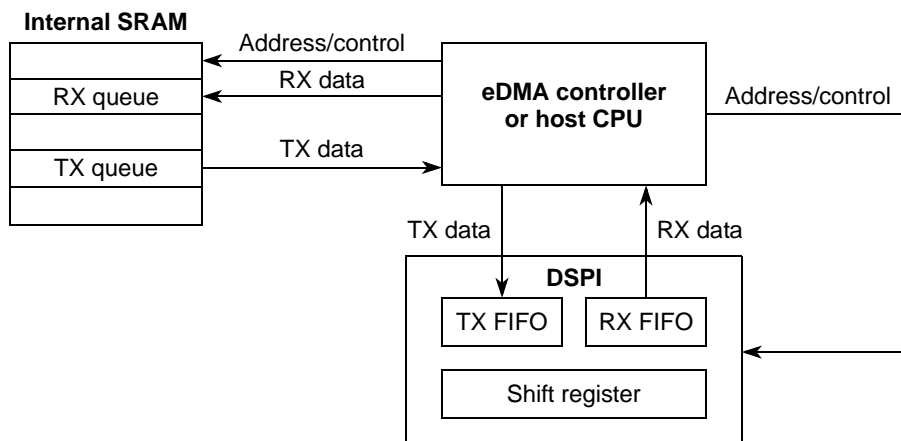


Figure 255. DSPI with queues and eDMA

24.4 Features

The DSPI supports these SPI features:

- Full-duplex, 3-wire synchronous transfers
- Master and slave mode
- Buffered transmit and receive operation using the TX and RX FIFOs, with depths of 5 entries
- Visibility into TX and RX FIFOs for ease of debugging
- FIFO bypass mode for low-latency updates to SPI queues
- Programmable transfer attributes on a per-frame basis
 - 8 clock and transfer attribute registers
 - Serial clock with programmable polarity and phase
 - Programmable delays
 - CS to SCK delay
 - SCK to CS delay

- Delay between frames
- Programmable serial frame size of 4 to 16 bits, expandable with software control
- Continuously held chip select capability
- 8 peripheral chip selects, expandable to 64 with external demultiplexer
- Deglitching support for as many as 32 peripheral chip selects with external demultiplexer
- 2 DMA conditions for SPI queues residing in RAM or flash
 - TX FIFO is not full (TFFF)
 - RX FIFO is not empty (RFDF)
- 7 interrupt conditions:
 - End of queue reached (EOQF)
 - TX FIFO is not full (TFFF)
 - Transfer of current frame complete (TCF)
 - RX FIFO is not empty (RFDF)
 - FIFO overrun (attempt to transmit with an empty TX FIFO or serial frame received while RX FIFO is full) (RFOF)
 - FIFO under flow (slave only and SPI mode, the slave is asked to transfer data when the TX FIFO is empty) (TFUF)
- Modified SPI transfer formats for communication with slower peripheral devices
- Continuous serial communications clock (SCK)

24.5 Modes of operation

The DSPI has four modes of operation. These modes can be divided into two categories; module-specific modes such as master, slave, and module disable modes, and a second category that is an MCU-specific mode: debug mode.

The module-specific modes are entered by host software writing to a register. The MCU-specific mode is controlled by signals external to the DSPI. The MCU-specific mode is a mode that the entire device may enter, in parallel to the DSPI being in one of its module-specific modes.

24.5.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK, \overline{CS}_n and SOUT signals are controlled by the DSPI and configured as outputs.

For more information, refer to [Section 24.8.1.1, “Master mode”](#).

24.5.2 Slave mode

Slave mode allows the DSPI to communicate with SPI bus masters. In this mode the DSPI responds to externally controlled serial transfers. The DSPI cannot initiate serial transfers in slave mode. In slave mode, the SCK signal and the CS0_x signal are configured as inputs and provided by a bus master. $\overline{CS0}_x$

must be configured as input and pulled high. If the internal pullup is being used then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE = 1], [WPS = 1]).

For more information, refer to [Section 24.8.1.2, “Slave mode”](#).

24.5.3 Module disable mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in module disable mode. The DSPI enters the module disable mode when the MDIS bit in DSPIx_MCR is set.

For more information, refer to [Section 24.8.1.3, “Module disable mode”](#).

24.5.4 Debug mode

Debug mode is used for system development and debugging. If the device enters debug mode while the FRZ bit in the DSPIx_MCR is set, the DSPI halts operation on the next frame boundary. If the device enters debug mode while the FRZ bit is cleared, the DSPI behavior is unaffected.

For more information, refer to [Section 24.8.1.4, “Debug mode”](#).

24.6 External signal description

24.6.1 Signal overview

[Table 284](#) lists off-chip DSPI signals.

Table 284. Signal properties

Name	I/O type	Function	
		Master mode	Slave mode
$\overline{\text{CS0}}_x$	Output / input	Peripheral chip select 0	Slave select
$\overline{\text{CS1:3}}_x$ (DSPI 0: $\overline{\text{CS1:3}}_0$, $\overline{\text{CS5}}_0$)	Output	Peripheral chip select 1–3	Unused ¹
$\overline{\text{CS4}}_x$ / MTRIG	Output	Peripheral chip select 4	Master trigger
$\overline{\text{CS5}}_x$ (DSPI 0: $\overline{\text{CS7}}_0$)	Output	Peripheral chip select 5 / Peripheral chip select strobe	Unused ¹
SIN _x	Input	Serial data in	Serial data in
SOUT _x	Output	Serial data out	Serial data out
SCK _x	Output / input	Serial clock (output)	Serial clock (input)

¹ The SIUL allows you to select alternate pin functions for the device.

24.6.2 Signal names and descriptions

24.6.2.1 Peripheral Chip Select / Slave Select (\overline{CS}_0)

In master mode, the \overline{CS}_0 signal is a peripheral chip select output that selects the slave device to which the current transmission is intended.

In slave mode, the \overline{CS}_0 signal is a slave select input signal that allows an SPI master to select the DSPI as the target for transmission. \overline{CS}_0 must be configured as input and pulled high. If the internal pullup is being used then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE = 1], [WPS = 1]).

Set the IBE and OBE bits in the PCR register for all \overline{CS}_0 pins when the DSPI chip select or slave select primary function is selected for that pin. When the pin is used for DSPI master mode as a chip select output, set the OBE bit. When the pin is used in DSPI slave mode as a slave select input, set the IBE bit.

Refer to [Section 13.5.3.8, “Pad Configuration Registers \(PCR\[0:70\]\)”](#), for more information.

24.6.2.2 Peripheral Chip Selects 1–3 ($\overline{CS1:3}$)

$\overline{CS1:3}$ are peripheral chip select output signals in master mode. In slave mode these signals are not used. On DSPI_0, these are $\overline{CS1:3}$ and $\overline{CS5:6}$.

24.6.2.3 Peripheral Chip Select 4 / Master Trigger ($\overline{CS4/MTRIG}$)

$\overline{CS4}$ is a peripheral chip select output signal in master mode. In slave mode this signal is not used.

24.6.2.4 Peripheral Chip Select 5/Peripheral Chip Select Strobe (\overline{CS}_5)

$\overline{CS5}$ is a peripheral chip select output signal. When the DSPI is in master mode and PCSSE bit in the DSPIx_MCR is cleared, the $\overline{CS5}$ signal selects the slave device that receives the current transfer.

$\overline{CS5}$ is a strobe signal used by external logic for deglitching of the CS signals. When the DSPI is in master mode and the PCSSE bit in the DSPIx_MCR is set, the $\overline{CS5}$ signal indicates the timing to decode $\overline{CS0:4}$ signals, which prevents glitches from occurring.

$\overline{CS5}$ is not used in slave mode. On DSPI_0, this is $\overline{CS7}$.

24.6.2.5 Serial Input (SIN_x)

SIN_x is a serial data input signal.

24.6.2.6 Serial Output ($SOUT_x$)

$SOUT_x$ is a serial data output signal.

24.6.2.7 Serial Clock (SCK_x)

SCK_x is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK_x is an input from an external bus master.

24.7 Memory map and registers description

24.7.1 Memory map

Table 285 shows the DSPI memory map.

Table 285. DSPI memory map

Offset from DSPI_BASE 0xFFF9_0000 (DSPI_0) 0xFFF9_4000 (DSPI_1) 0xFFF9_8000 (DSPI_2)	Register	Access	Reset value	Location
0x0000	DSPI_MCR—DSPI module configuration register	R/W	0x0000_0001	on page 521
0x0004	DSPI Hardware Configuration Register (DSPI_HCR) ¹			
0x0008	DSPI_TCR—DSPI transfer count register	R/W	0x0000_0000	on page 525
0x000C	DSPI_CTAR0—DSPI clock and transfer attributes register 0	R/W	0x0000_0000	on page 525
0x0010	DSPI_CTAR1—DSPI clock and transfer attributes register 1	R/W	0x0000_0000	on page 525
0x0014	DSPI_CTAR2—DSPI clock and transfer attributes register 2	R/W	0x0000_0000	on page 525
0x0018	DSPI_CTAR3—DSPI clock and transfer attributes register 3	R/W	0x0000_0000	on page 525
0x001C	DSPI_CTAR4—DSPI clock and transfer attributes register 4	R/W	0x0000_0000	on page 525
0x0020	DSPI_CTAR5—DSPI clock and transfer attributes register 5	R/W	0x0000_0000	on page 525
0x0024	DSPI_CTAR6—DSPI clock and transfer attributes register 6	R/W	0x0000_0000	on page 525
0x0028	DSPI_CTAR7—DSPI clock and transfer attributes register 7	R/W	0x0000_0000	on page 525
0x002C	DSPI_SR—DSPI status register	R/W	0x0000_0000	on page 531
0x0030	DSPI_RSER—DSPI DMA/interrupt request select and enable register	R/W	0x0000_0000	on page 533
0x0034	DSPI_PUSHR—DSPI push TX FIFO register	R/W	0x0000_0000	on page 534

Table 285. DSPI memory map (continued)

Offset from DSPI_BASE 0xFFF9_0000 (DSPI_0) 0xFFF9_4000 (DSPI_1) 0xFFF9_8000 (DSPI_2)	Register	Access	Reset value	Location
0x0038	DSPI_POPR—DSPI pop RX FIFO register	R	0x0000_0000	on page 536
0x003C	DSPI_TXFR0—DSPI transmit FIFO register 0	R	0x0000_0000	on page 537
0x0040	DSPI_TXFR1—DSPI transmit FIFO register 1	R	0x0000_0000	on page 537
0x0044	DSPI_TXFR2—DSPI transmit FIFO register 2	R	0x0000_0000	on page 537
0x0048	DSPI_TXFR3—DSPI transmit FIFO register 3	R	0x0000_0000	on page 537
0x004C	DSPI_TXFR4—DSPI transmit FIFO register 4	R	0x0000_0000	on page 537
0x0050–0x007B	Reserved			
0x007C	DSPI_RXFR0—DSPI receive FIFO register 0	R	0x0000_0000	on page 537
0x0080	DSPI_RXFR1—DSPI receive FIFO register 1	R	0x0000_0000	on page 537
0x0084	DSPI_RXFR2—DSPI receive FIFO register 2	R	0x0000_0000	on page 537
0x0088	DSPI_RXFR3—DSPI receive FIFO register 3	R	0x0000_0000	on page 537
0x008C	DSPI_RXFR4—DSPI receive FIFO register 4	R	0x0000_0000	on page 537
0x0090–0x3FFF	Reserved			

¹ The DSPI_HCR register provides parametrization information about particular instance of the DSPI module.

24.7.2 Registers description

24.7.2.1 DSPI Module Configuration Register (DSPIx_MCR)

The DSPIx_MCR contains bits that configure attributes of the DSPI operation. The values of the HALT and MDIS bits can be changed at any time, but their effect begins on the next frame boundary. The HALT and MDIS bits in the DSPIx_MCR are the only bit values software can change while the DSPI is running.

Deserial Serial Peripheral Interface (DSPI)

Address: Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	DCONF[0:1]		FRZ	MTFE	PCSSE	ROOE	PCSIS7	PCSIS6	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MDIS	DIS_TXF	DIS_RXF	CLR_TXF	CLR_RXF	SMPL_PT[0:1]		0	0	0	0	0	FCPCS	0	HALT
W					w1c	w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 256. DSPI Module Configuration Register (DSPIx_MCR)

Table 286. DSPIx_MCR field descriptions

Field	Description										
MSTR	Master/slave mode select Configures the DSPI for master mode or slave mode. 0 DSPI is in slave mode. 1 DSPI is in master mode.										
CONT_SCKE	Continuous SCK enable Enables the serial communication clock (SCK) to run continuously. Refer to Section 24.8.6, “Continuous Serial communications clock” , for details. 0 Continuous SCK disabled 1 Continuous SCK enabled										
DCONF [0:1]	DSPI configuration The following table lists the DCONF values for the various configurations. <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>DCONF</th> <th>Configuration</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>SPI</td> </tr> <tr> <td>01</td> <td>Invalid value</td> </tr> <tr> <td>10</td> <td>Invalid value</td> </tr> <tr> <td>11</td> <td>Invalid value</td> </tr> </tbody> </table>	DCONF	Configuration	00	SPI	01	Invalid value	10	Invalid value	11	Invalid value
DCONF	Configuration										
00	SPI										
01	Invalid value										
10	Invalid value										
11	Invalid value										
FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters debug mode. 0 Do not halt serial transfers. 1 Halt serial transfers.										
MTFE	Modified timing format enable Enables a modified transfer format to be used. Refer to Section 24.8.5.4, “Modified SPI transfer format (MTFE = 1, CPHA = 1)” , for more information. 0 Modified SPI transfer format disabled 1 Modified SPI transfer format enabled										

Table 286. DSPIx_MCR field descriptions (continued)

Field	Description
PCSSE	Peripheral chip select strobe enable _____ Enables the CS5_x to operate as a CS strobe output signal. Refer to Section 24.8.4.5, “Peripheral Chip Select strobe enable (CS5_x)” , for more information. 0 CS5_x is used as the Peripheral chip select 5 signal. 1 CS5_x is used as an active-low CS strobe signal.
ROOE	Receive FIFO overflow overwrite enable Enables an RX FIFO overflow condition to ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer that generated the overflow is ignored or put in the shift register. If the ROOE bit is set, the incoming data is put in the shift register. If the ROOE bit is cleared, the incoming data is ignored. Refer to Section 24.8.7.6, “Receive FIFO overflow interrupt request (RFOF)” , for more information. 0 Incoming data is ignored. 1 Incoming data is put in the shift register.
PCSiSn	Peripheral chip select inactive state _____ Determines the inactive state of the CS0_x signal. CS0_x must be configured as inactive high for slave mode operation. 0 The inactive state of CS0_x is low. 1 The inactive state of CS0_x is high. Note: PCSiS7 and PCSiS6 are implemented only on DSPI_0.
MDIS	Module disable Allows the clock to stop to the non-memory mapped logic in the DSPI, effectively putting the DSPI in a software controlled power-saving state. Refer to Section 24.8.8, “Power saving features” , for more information.” The reset value of the MDIS bit is parameterized, with a default reset value of 0. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
DIS_TXF	Disable transmit FIFO Enables and disables the TX FIFO. When the TX FIFO is disabled, the transmit part of the DSPI operates as a simplified double-buffered SPI. Refer to Section 24.8.3.3, “FIFO disable operation” , for details. 0 TX FIFO enabled 1 TX FIFO disabled
DIS_RXF	Disable receive FIFO Enables and disables the RX FIFO. When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. Refer to Section 24.8.3.3, “FIFO disable operation” , for details. 0 RX FIFO enabled 1 RX FIFO disabled
CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Write a 1 to the CLR_TXF bit to clear the TX FIFO counter. The CLR_TXF bit is always read as 0. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
CLR_RXF	Clear RX FIFO Flushes the RX FIFO. Write a 1 to the CLR_RXF bit to clear the RX counter. The CLR_RXF bit is always read as 0. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.

Table 286. DSPIx_MCR field descriptions (continued)

Field	Description										
SMPL_PT [0:1]	<p>Sample point Allows the host software to select when the DSPI master samples SIN in modified transfer format. Figure 273 shows where the master can sample the SIN pin. The following table lists the delayed sample points.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>SMPL_PT</th> <th>Number of system clock cycles between odd-numbered edge of SCK_x and sampling of SIN_x</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>0</td> </tr> <tr> <td>01</td> <td>1</td> </tr> <tr> <td>10</td> <td>2</td> </tr> <tr> <td>11</td> <td>Invalid value</td> </tr> </tbody> </table>	SMPL_PT	Number of system clock cycles between odd-numbered edge of SCK_x and sampling of SIN_x	00	0	01	1	10	2	11	Invalid value
SMPL_PT	Number of system clock cycles between odd-numbered edge of SCK_x and sampling of SIN_x										
00	0										
01	1										
10	2										
11	Invalid value										
FCPCS	<p>Fast Continuous PCS Mode. This bit enables the masking of “After SCK (t_{ASC})” & “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits 6 & 7 of the DSPI_PUSHR register. The firmware should select appropriate masks when providing continuous frames via the DSPI_PUSHR register.</p> <p>0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in DSPI_PUSHR register.</p>										
HALT	<p>Halt Provides a mechanism for software to start and stop DSPI transfers. Refer to Section 24.8.2, “Start and stop of DSPI transfers”, for details on the operation of this bit.</p> <p>0 Start transfers. 1 Stop transfers.</p>										

24.7.2.2 DSPI Hardware Configuration Register (DSPI_HCR)

DSPI Hardware Configuration Register provides particular implementation details about the DSPI module, i.e. number of Receive and Transmit FIFO entries, number of CTAR registers and if DSI features are implemented in the module or not. It is read only register.

Address: DSPI_BASE + 0x4

Access:

	R	DSI	0	0	0	0	0	CTAR	TXFR	RXFR								
	W																	
Reset		0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	1	1
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																	
Reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 257. DSPI Hardware Configuration Register (DSPI_HCR)

Table 287. DSPI_HCR Field Descriptions

Field	Description
DSI	DSI features are implemented for the module. 0 - DSI features are not implemented, DSI registers don't exist. 1 - DSI features are implemented
CTAR[]	CTAR, Maximum implemented DSPI_CTAR register number.
TXFR	TXFR, Maximum implemented DSPI_TXFR register number.
RXFR[]	RXFR, Maximum implemented DSPI_RXFR register number.

24.7.2.3 DSPI Transfer Count Register (DSPIx_TCR)

The DSPIx_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. The user must not write to the DSPIx_TCR while the DSPI is running.

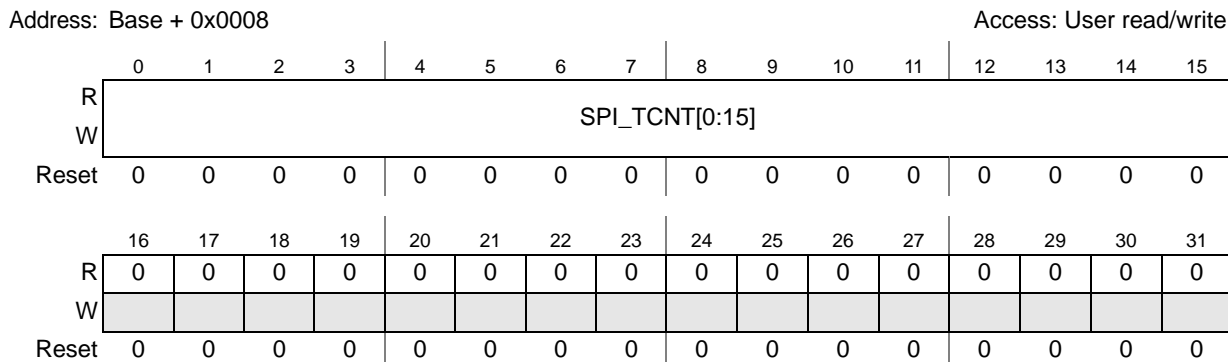


Figure 258. DSPI Transfer Count Register (DSPIx_TCR)

Table 288. DSPIx_TCR field descriptions

Field	Description
SPI_TCNT [0:15]	SPI transfer counter Counts the number of SPI transfers the DSPI makes. The SPI_TCNT field is incremented every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The transfer counter 'wraps around,' incrementing the counter past 65535 resets the counter to zero.

24.7.2.4 DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn)

The DSPI modules each contain eight clock and transfer attribute registers (DSPIx_CTARn) that define different transfer attribute configurations. Each DSPIx_CTAR controls:

- Frame size
- Baud rate and transfer delay values
- Clock phase
- Clock polarity

- MSB or LSB first

DSPI_x_CTARs support compatibility with the QSPI module used in certain members of the MPC56x family of MCUs. At the initiation of an SPI transfer, control logic selects the DSPI_x_CTAR that contains the transfer's attributes. Do not write to the DSPI_x_CTARs while the DSPI is running.

In master mode, the DSPI_x_CTAR_n registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bit fields in the DSPI_x_CTAR0 and DSPI_x_CTAR1 registers sets the slave transfer attributes. Refer to the individual bit descriptions for details on which bits are used in slave modes.

When the DSPI is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the DSPI_x_CTAR registers is used on a per-frame basis. When the DSPI is configured as an SPI bus slave, the DSPI_x_CTAR0 register is used.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks with help of PDT and DT fields.

Address: DSPI_BASE + 0xC–DSPI_BASE + 0x28

Access:

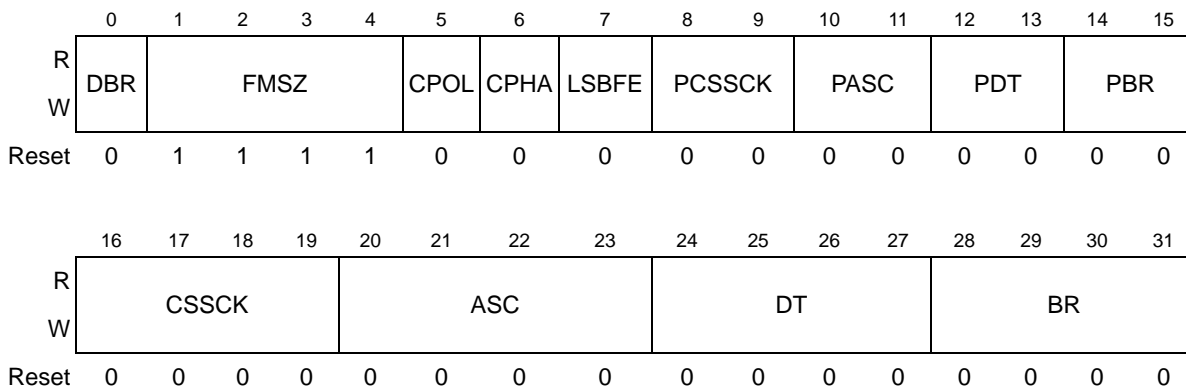


Figure 259. DSPI Clock and Transfer Attributes Register 0–7 (DSPI_CTAR0–DSPI_CTAR7) in the master mode

Address: DSPI_BASE + 0xC /0x10

Access:

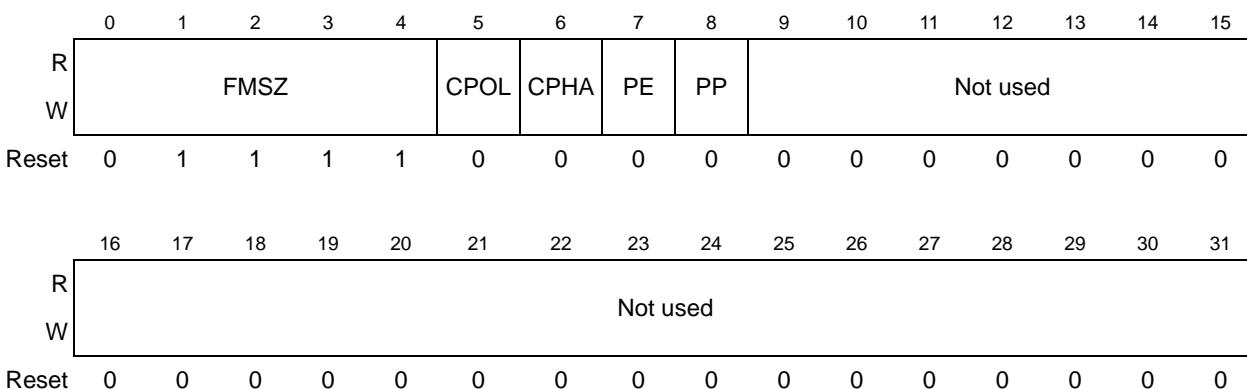


Figure 260. DSPI Clock and Transfer Attributes Register 0, 1 (DSPI_CTAR0, DSPI_CTAR1) in the slave mode

Table 289. DSPI_CTAR_n Field Descriptions in master mode

Field	Descriptions
DBR	<p>Double Baud Rate. The DBR bit doubles the effective baud rate of the Serial Communications Clock (SCK). This field is only used in master mode. It effectively halves the Baud Rate division ratio supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit. See the BR field description for details on how to compute the baud rate.</p> <p>0 The baud rate is computed normally with a 50/50 duty cycle 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler</p>
FMSZ[0:3]	<p>Frame Size. The number of bits transferred per frame is equal FMSZ field value plus 1. Minimum valid FMSZ field value is 3.</p> <p>When operating in TSB mode, the FMSZ field value plus 1 is equal the data frame bit number, where control of the PCS assertion switches from the DSPI_DSICR to the DSPI_DSICR1 register.</p>
CPOL	<p>Clock Polarity. The CPOL bit selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge.</p> <p>0 The inactive state value of SCK is low 1 The inactive state value of SCK is high</p>
CPHA	<p>Clock Phase. The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode the bit value is ignored and the transfers are done as CPHA bit is set to 1.</p> <p>0 Data is captured on the leading edge of SCK and changed on the following edge 1 Data is changed on the leading edge of SCK and captured on the following edge</p>
LSBFE	<p>LSB First. The LSBFE bit selects if the LSB or MSB of the frame is transferred first. When operating in TSB configuration, this bit should be set to be compliant to MSC specification.</p> <p>0 Data is transferred MSB first 1 Data is transferred LSB first</p>

Table 289. DSPI_CTAR_n Field Descriptions in master mode

Field	Descriptions
PCSSCK[0:1]	<p>PCS to SCK Delay Prescaler. The PCSSCK field selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description how to compute the PCS to SCK Delay. In the TSB mode the PCSSCK field has no effect.</p> <p>00 PCS to SCK Prescaler value is 1 01 PCS to SCK Prescaler value is 3 10 PCS to SCK Prescaler value is 5 11 PCS to SCK Prescaler value is 7</p>
PASC[0:1]	<p>After SCK Delay Prescaler. The PASC field selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description how to compute the After SCK Delay. In the TSB mode the PASC field has no effect.</p> <p>00 After SCK Delay Prescaler value is 1 01 After SCK Delay Prescaler value is 3 10 After SCK Delay Prescaler value is 5 11 After SCK Delay Prescaler value is 7</p>
PDT[0:1]	<p>Delay after Transfer Prescaler. The PDT field selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. In the TSB mode the PDT field defines two MSB bits of the Delay after Transfer. See the DT field description for details on how to compute the Delay after Transfer.</p> <p>00 Delay after Transfer Prescaler value is 1 01 Delay after Transfer Prescaler value is 3 10 Delay after Transfer Prescaler value is 5 11 Delay after Transfer Prescaler value is 7</p>
PBR[0:1]	<p>Baud Rate Prescaler. The PBR field selects the prescaler value for the baud rate. This field is only used in master mode. The Baud Rate is the frequency of the Serial Communications Clock (SCK). The system clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate.</p> <p>00 Baud Rate Prescaler value is 2 01 Baud Rate Prescaler value is 3 10 Baud Rate Prescaler value is 5 11 Baud Rate Prescaler value is 7</p>

Table 289. DSPI_CTAR_n Field Descriptions in master mode

Field	Descriptions
CSSCK[0:3]	<p>PCS to SCK Delay Scaler. The CSSCK field selects the scaler value for the PCS to SCK delay. This field is only used in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The PCS to SCK Delay is a multiple of the system clock period and it is computed according to the following equation:</p> $t_{CSC} = \frac{1}{f_{SYS}} \times PCSSCK \times CSSCK \quad \text{Eqn. 1}$ <p>See Section 24.8.4.2, "CS to SCK delay (tCSC)", for more details. In the TSB mode the field has no effect.</p>
ASC[0:3]	<p>After SCK Delay Scaler. The ASC field selects the scaler value for the After SCK Delay. This field is only used in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The After SCK Delay is a multiple of the system clock period, and it is computed according to the following equation:</p> $t_{ASC} = \frac{1}{f_{SYS}} \times PASC \times ASC \quad \text{Eqn. 2}$ <p>See Section 24.8.4.3, "After SCK delay (tASC)", for more details. In the TSB mode the field has no effect.</p>
DT[0:3]	<p>Delay after Transfer Scaler. The DT field selects the Delay after Transfer Scaler. This field is only used in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. . In the Continuous Serial Communications Clock operation the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the system clock period and it is computed according to the following equation:</p> $t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT \quad \text{Eqn. 3}$ <p>In the TSB mode the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See Section 24.8.4.4, "Delay after transfer (tDT)", for more details.</p>
BR[0:3]	<p>Baud Rate Scaler. The BR field selects the scaler value for the baud rate. This field is only used in master mode. The prescaled system clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation:</p> $\text{SCK baud rate} = \frac{f_{SYS}}{PBR} \times \frac{1 + DBR}{BR} \quad \text{Eqn. 4}$ <p>See Section 24.8.4.1, "Baud rate generator", for more details.</p>

Table 290. DSPI SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

Table 291. Delay Scaler Encoding

Field value	Scaler Value	Field value	Scaler Value
0000	2	1000	512
0001	4	1001	1024
0010	8	1010	2048
0011	16	1011	4096
0100	32	1100	8192
0101	64	1101	16384
0110	128	1110	32768
0111	256	1111	65536

Table 292. DSPI Baud Rate Scaler

BR	Baud Rate Scaler Value	BR	Baud Rate Scaler Value
0000	2	1000	256
0001	4	1001	512
0010	6	1010	1024
0011	8	1011	2048
0100	16	1100	4096
0101	32	1101	8192
0110	64	1110	16384
0111	128	1111	32768

Table 293. DSPI_CTAR0, DSPI_CTAR1 Field Descriptions in slave mode

Field	Descriptions
FMSZ[0:4]	Frame Size. The number of bits transferred per frame is equal FMSZ field value plus 1. Minimum valid FMSZ field value is 3.
CPOL	Clock Polarity. The CPOL bit selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low 1 The inactive state value of SCK is high
CPHA	Clock Phase. The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. 0 Data is captured on the leading edge of SCK and changed on the following edge 1 Data is changed on the leading edge of SCK and captured on the following edge
PE	Parity Enable. PE bit enables parity bit transmission and reception for the frame 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
PP	Parity Polarity. PP bit controls polarity of the parity bit transmitted and checked 0 Even Parity: number of "1" bits in the transmitted frame is even. The DSPI_SR[SPEF] bit is set if in the received frame number of "1" bits is odd. 1 Odd Parity: number of "1" bits in the transmitted frame is odd. The DSPI_SR[SPEF] bit is set if in the received frame number of "1" bits is even.
29–31 -	Not used, write always zero to keep software compatible with future updates.

24.7.2.5 DSPI Status Register (DSPIx_SR)

The DSPIx_SR contains status and flag bits. The bits are set by the hardware and reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the DSPIx_SR by writing a 1 to clear it (w1c). Writing a 0 to a flag bit has no effect.

Address: Base + 0x002C

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	0
W	w1c			w1c	w1c		w1c						w1c		w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 261. DSPI Status Register (DSPIx_SR)

Table 294. DSPIx_SR field descriptions

Field	Description
TCF	<p>Transfer complete flag</p> <p>Indicates that all bits in a frame have been shifted out. The TCF bit is set after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 24.8.5.1, “Classic SPI transfer format (CPHA = 0)”, for details. The TCF bit is cleared by writing 1 to it.</p> <p>0 Transfer not complete. 1 Transfer complete.</p>
TXRXS	<p>TX and RX status</p> <p>Reflects the status of the DSPI. Refer to Section 24.8.2, “Start and stop of DSPI transfers”, for information on what clears and sets this bit.</p> <p>0 TX and RX operations are disabled (DSPI is in STOPPED state). 1 TX and RX operations are enabled (DSPI is in RUNNING state).</p>
EOQF	<p>End of queue flag</p> <p>Indicates that transmission in progress is the last entry in a queue. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 24.8.5.1, “Classic SPI transfer format (CPHA = 0)”, for details.</p> <p>The EOQF bit is cleared by writing 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared.</p> <p>0 EOQ is not set in the executing command. 1 EOQ bit is set in the executing SPI command.</p> <p>Note: EOQF does not function in slave mode.</p>
TFUF	<p>Transmit FIFO underflow flag</p> <p>Indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in SPI slave mode is empty, and a transfer is initiated by an external SPI master. The TFUF bit is cleared by writing 1 to it.</p> <p>0 TX FIFO underflow has not occurred. 1 TX FIFO underflow has occurred.</p>
TFFF	<p>Transmit FIFO fill flag</p> <p>Indicates that the TX FIFO can be filled. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it, or an by acknowledgement from the eDMA controller when the TX FIFO is full.</p> <p>0 TX FIFO is full. 1 TX FIFO is not full.</p>
RFOF	<p>Receive FIFO overflow flag</p> <p>Indicates that an overflow condition in the RX FIFO has occurred. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit is cleared by writing 1 to it.</p> <p>0 RX FIFO overflow has not occurred. 1 RX FIFO overflow has occurred.</p>
RFDF	<p>Receive FIFO drain flag</p> <p>Indicates that the RX FIFO can be drained. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it, or by acknowledgement from the eDMA controller when the RX FIFO is empty.</p> <p>0 RX FIFO is empty. 1 RX FIFO is not empty.</p> <p>Note: In the interrupt service routine, RFDF must be cleared only after the DSPIx_POPR register is read.</p>

Table 294. DSPIx_SR field descriptions (continued)

Field	Description
TXCTR [0:3]	TX FIFO counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the DSPI_PUSH is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
TXNXPTR [0:3]	Transmit next pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. Refer to Section 24.8.3.4, "Transmit First In First Out (TX FIFO) buffering mechanism" , for more details.
RXCTR [0:3]	RX FIFO counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the DSPI_POP is read. The RXCTR is incremented after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 24.8.5.1, "Classic SPI transfer format (CPHA = 0)" , for details.
POPXPTR [0:3]	Pop next pointer Contains a pointer to the RX FIFO entry that is returned when the DSPIx_POP is read. The POPXPTR is updated when the DSPIx_POP is read. Refer to Section 24.8.3.5, "Receive First In First Out (RX FIFO) buffering mechanism" , for more details.

24.7.2.6 DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER)

The DSPIx_RSER serves two purposes: enables flag bits in the DSPIx_SR to generate DMA requests or interrupt requests, and selects the type of request to generate. Refer to the bit descriptions for the type of requests that are supported. Do not write to the DSPIx_RSER while the DSPI is running.

Address: Base + 0x0030

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	0	0	EOQF_RE	TFUF_RE	0	TFFF_RE	TFFF_DIRS	0	0	0	0	RFOF_RE	0	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 262. DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER)
Table 295. DSPIx_RSER field descriptions

Field	Description
TCF_RE	Transmission complete request enable Enables TCF flag in the DSPIx_SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.

Table 295. DSPIx_RSER field descriptions (continued)

Field	Description
EOQF_RE	DSPI finished request enable Enables the EOQF flag in the DSPIx_SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
TFUF_RE	Transmit FIFO underflow request enable The TFUF_RE bit enables the TFUF flag in the DSPIx_SR to generate an interrupt request. 0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
TFFF_RE	Transmit FIFO fill request enable Enables the TFFF flag in the DSPIx_SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA requests. 0 TFFF interrupt requests or DMA requests are disabled. 1 TFFF interrupt requests or DMA requests are enabled.
TFFF_DIRS	Transmit FIFO fill DMA or interrupt request select Selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the DSPIx_SR is set, and the TFFF_RE bit in the DSPIx_RSER is set, this bit selects between generating an interrupt request or a DMA request. 0 Interrupt request is selected. 1 DMA request is selected.
RFOF_RE	Receive FIFO overflow request enable Enables the RFOF flag in the DSPIx_SR to generate an interrupt requests. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
RFDF_RE	Receive FIFO drain request enable Enables the RFDF flag in the DSPIx_SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt requests or DMA requests are disabled. 1 RFDF interrupt requests or DMA requests are enabled.
RFDF_DIRS	Receive FIFO drain DMA or interrupt request select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the DSPIx_SR is set, and the RFDF_RE bit in the DSPIx_RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request is selected. 1 DMA request is selected.

24.7.2.7 DSPI PUSH TX FIFO Register (DSPIx_PUSHR)

The DSPIx_PUSHR provides a means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. Refer to [Section 24.8.3.4, “Transmit First In First Out \(TX FIFO\) buffering mechanism”](#), for more information. Write accesses of 8 or 16 bits to the DSPIx_PUSHR transfer 32 bits to the TX FIFO.

NOTE

TXDATA is used in master and slave modes.

Address Base + 0x0034

Access: User read/write

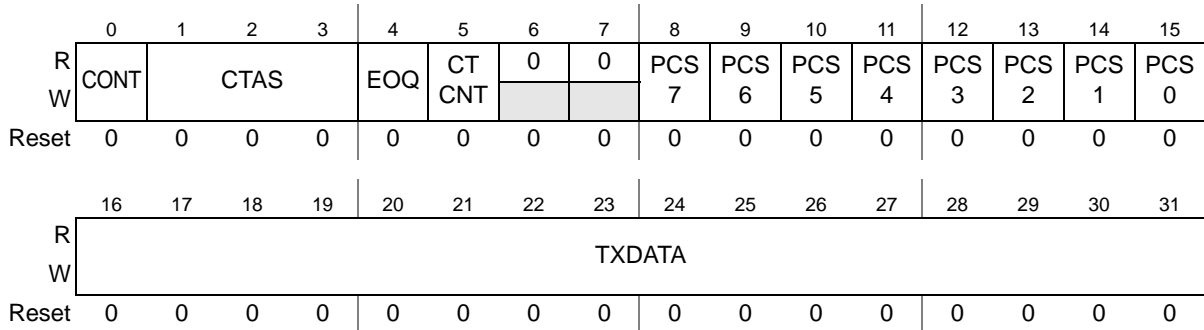


Figure 263. DSPI PUSH TX FIFO Register (DSPIx_PUSHR)

Table 296. DSPIx_PUSHR field descriptions

Field	Description																		
CONT	<p>Continuous peripheral chip select enable</p> <p>Selects a continuous selection format. The bit is used in SPI master mode. The bit enables the selected CS signals to remain asserted between transfers. Refer to Section 24.8.5.5, “Continuous selection format”, for more information.</p> <p>0 Return peripheral chip select signals to their inactive state between transfers. 1 Keep peripheral chip select signals asserted between transfers.</p>																		
CTAS [0:2]	<p>Clock and transfer attributes select</p> <p>Selects which of the DSPIx_CTARs sets the transfer attributes for the SPI frame. In SPI slave mode, DSPIx_CTAR0 is used. The following table shows how the CTAS values map to the DSPIx_CTARs. There are eight DSPIx_CTARs in the device DSPI implementation.</p> <p>Note: Use in SPI master mode only.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>CTAS</th> <th>Use Clock and Transfer Attributes from</th> </tr> </thead> <tbody> <tr><td>000</td><td>DSPIx_CTAR0</td></tr> <tr><td>001</td><td>DSPIx_CTAR1</td></tr> <tr><td>010</td><td>DSPIx_CTAR2</td></tr> <tr><td>011</td><td>DSPIx_CTAR3</td></tr> <tr><td>100</td><td>DSPIx_CTAR4</td></tr> <tr><td>101</td><td>DSPIx_CTAR5</td></tr> <tr><td>110</td><td>DSPIx_CTAR6</td></tr> <tr><td>111</td><td>DSPIx_CTAR7</td></tr> </tbody> </table>	CTAS	Use Clock and Transfer Attributes from	000	DSPIx_CTAR0	001	DSPIx_CTAR1	010	DSPIx_CTAR2	011	DSPIx_CTAR3	100	DSPIx_CTAR4	101	DSPIx_CTAR5	110	DSPIx_CTAR6	111	DSPIx_CTAR7
CTAS	Use Clock and Transfer Attributes from																		
000	DSPIx_CTAR0																		
001	DSPIx_CTAR1																		
010	DSPIx_CTAR2																		
011	DSPIx_CTAR3																		
100	DSPIx_CTAR4																		
101	DSPIx_CTAR5																		
110	DSPIx_CTAR6																		
111	DSPIx_CTAR7																		
EOQ	<p>End of queue</p> <p>Provides a means for host software to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer the EOQF bit in the DSPIx_SR is set.</p> <p>0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.</p> <p>Note: Use in SPI master mode only.</p>																		

Table 296. DSPIx_PUSHR field descriptions (continued)

Field	Description
CTCNT	<p>Clear SPI_TCNT</p> <p>Provides a means for host software to clear the SPI transfer counter. The CTCNT bit clears the SPI_TCNT field in the DSPIx_TCR. The SPI_TCNT field is cleared before transmission of the current SPI frame begins.</p> <p>0 Do not clear SPI_TCNT field in the DSPIx_TCR. 1 Clear SPI_TCNT field in the DSPIx_TCR.</p> <p>Note: Use in SPI master mode only.</p>
PCSx	<p>Peripheral chip select x</p> <p>Selects which \overline{CSx} signals are asserted for the transfer.</p> <p>0 Negate the \overline{CSx} signal. 1 Assert the \overline{CSx} signal.</p> <p>Note: Use in SPI master mode only.</p>
TXDATA [0:15]	<p>Transmit data</p> <p>Holds SPI data for transfer according to the associated SPI command.</p> <p>Note: Use TXDATA in master and slave modes.</p>

24.7.2.8 DSPI POP RX FIFO Register (DSPIx_POPR)

The DSPIx_POPR allows you to read the RX FIFO. Refer to [Section 24.8.3.5, “Receive First In First Out \(RX FIFO\) buffering mechanism”](#), for a description of the RX FIFO operations. Eight or 16-bit read accesses to the DSPIx_POPR fetch the RX FIFO data, and update the counter and pointer.

NOTE

Reading the RX FIFO field fetches data from the RX FIFO. Once the RX FIFO is read, the read data pointer is moved to the next entry in the RX FIFO. Therefore, read DSPIx_POPR only when you need the data. For compatibility, configure the TLB (MMU table) entry for DSPIx_POPR as guarded.

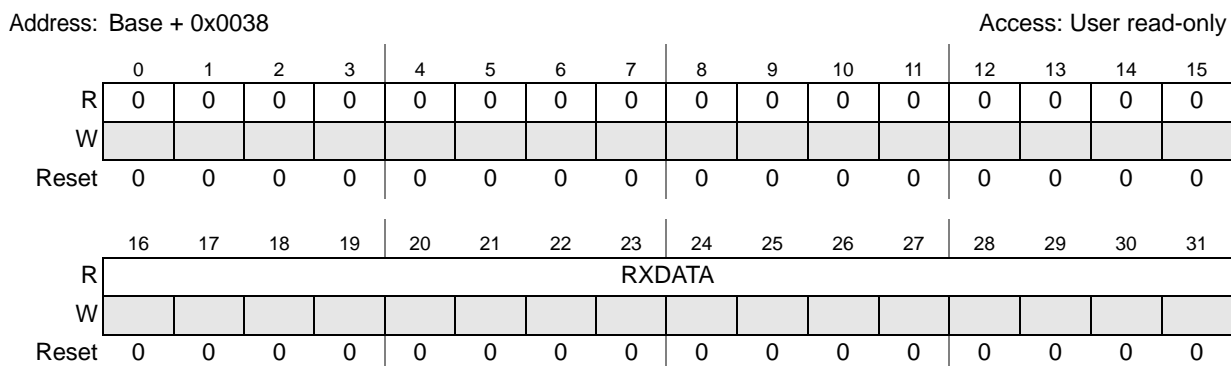


Figure 264. DSPI POP RX FIFO Register (DSPIx_POPR)

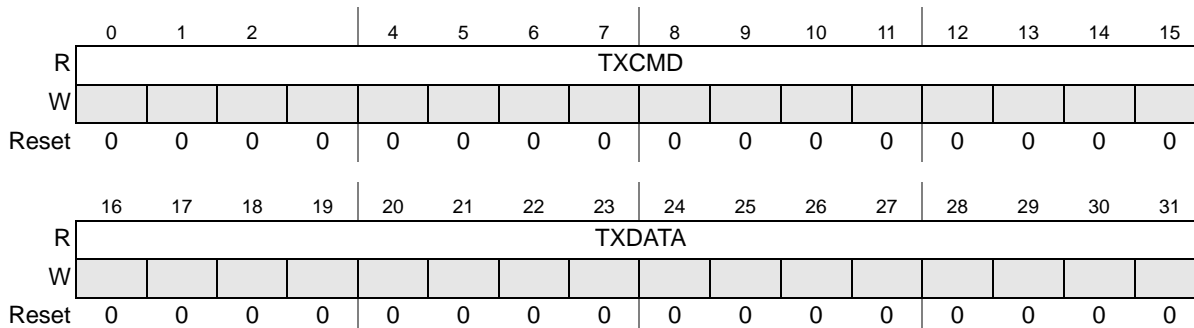
Table 297. DSPI_x_POPR field descriptions

Field	Description
RXDATA [0:15]	Received data The RXDATA field contains the SPI data from the RX FIFO entry pointed to by the pop next data pointer (POPNEXTPTR).

24.7.2.9 DSPI Transmit FIFO Registers 0–4 (DSPI_x_TXFR_n)

The DSPI_x_TXFR_n registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the DSPI_x_TXFR_n registers does not alter the state of the TX FIFO. The MCU uses five registers to implement the TX FIFO, that is DSPI_x_TXFR0–DSPI_x_TXFR4 are used.

Address: Base + 0x003C (DSPI_x_TXFR0) Base + 0x0048 (DSPI_x_TXFR3) Access: User read-only
 Base + 0x0040 (DSPI_x_TXFR1) Base + 0x004C (DSPI_x_TXFR4)
 Base + 0x0044 (DSPI_x_TXFR2)


Figure 265. DSPI Transmit FIFO Register 0–4 (DSPI_x_TXFR_n)
Table 298. DSPI_x_TXFR_n field descriptions

Field	Description
TXCMD [0:15]	Transmit command Contains the command that sets the transfer attributes for the SPI data. Refer to Section 24.7.2.7, “DSPI PUSH TX FIFO Register (DSPI_x_PUSHR)” , for details on the command field.
TXDATA [0:15]	Transmit data Contains the SPI data to be shifted out.

24.7.2.10 DSPI Receive FIFO Registers 0–4 (DSPI_x_RXFR_n)

The DSPI_x_RXFR_n registers provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The DSPI_x_RXFR registers are read-only. Reading the DSPI_x_RXFR_n registers does not alter the state of the RX FIFO. The device uses five registers to implement the RX FIFO, that is DSPI_x_RXFR0–DSPI_x_RXFR4 are used.

Address: Base + 0x007C (DSPIx_RXFR0) Base + 0x0088 (DSPIx_RXFR3) Access: User read-only
 Base + 0x0080 (DSPIx_RXFR1) Base + 0x008C (DSPIx_RXFR4)
 Base + 0x0084 (DSPIx_RXFR2)

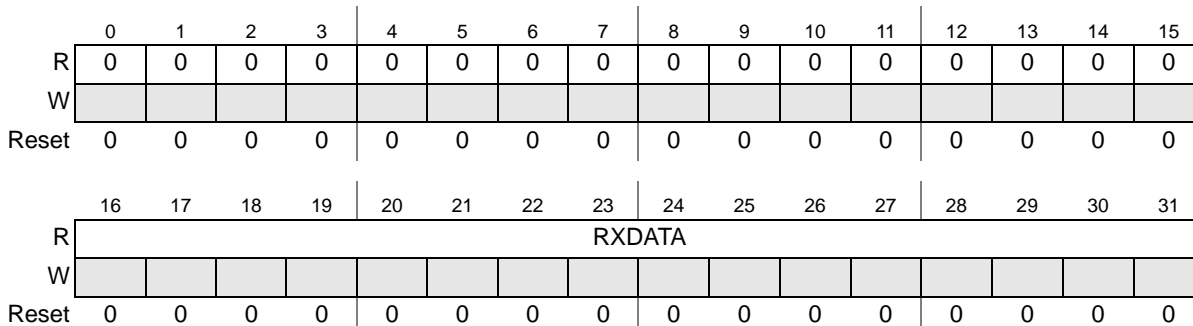


Figure 266. DSPI Receive FIFO Registers 0–4 (DSPIx_RXFRn)

Table 299. DSPIx_RXFRn field description

Field	Description
RXDATA [15:0]	Receive data Contains the received SPI data.

24.8 Functional description

The DSPI supports full-duplex, synchronous serial communications between the MCU and peripheral devices. All communications are through an SPI-like protocol.

The DSPI supports only the serial peripheral interface (SPI) configuration in which the DSPI operates as a basic SPI or a queued SPI.

The DCONF field in the DSPIx_MCR register determines the DSPI configuration. Refer to [Table 286](#) for the DSPI configuration values.

The DSPIx_CTAR0–DSPIx_CTAR7 registers hold clock and transfer attributes. The SPI configuration can select which CTAR to use on a frame by frame basis by setting the CTAS field in the DSPIx_PUSHR.

The 16-bit shift register in the master and the 16-bit shift register in the slave are linked by the SOUT_x and SIN_x signals to form a distributed 32-bit register. When a data transfer operation is performed, data is serially shifted a pre-determined number of bit positions. Because the registers are linked, data is exchanged between the master and the slave; the data that was in the master’s shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the TCF bit in the DSPIx_SR is set to indicate a completed transfer. [Figure 267](#) illustrates how master and slave data is exchanged.

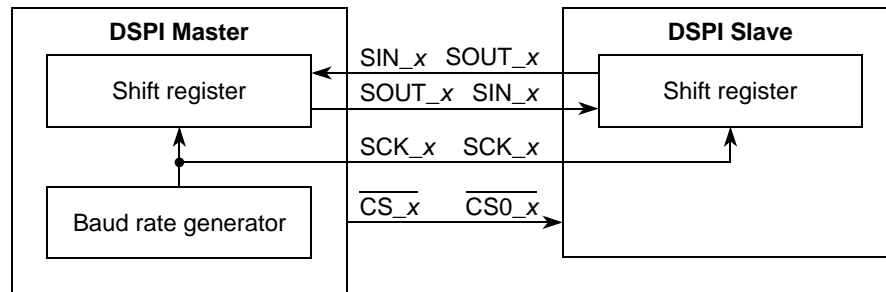


Figure 267. SPI serial protocol overview

Each DSPI has peripheral chip select (\overline{CS}_x) signals that select the slaves with which to communicate (DSPI_0 has eight \overline{CS}_x signals.)

Transfer protocols and timing properties are shared by the three DSPI configurations; these properties are described independently of the configuration in [Section 24.8.5, “Transfer formats”](#). The transfer rate and delay settings are described in [Section 24.8.4, “DSPI baud rate and clock delay generation”](#).

Refer to [Section 24.8.8, “Power saving features”](#), for information on the power-saving features of the DSPI.

24.8.1 Modes of operation

The DSPI modules have four available distinct modes:

- Master mode
- Slave mode
- Module disable mode
- Debug mode

Master, slave, and module disable modes are module-specific modes while debug mode is a device-specific mode. All four modes are implemented on this device.

The module-specific modes are determined by bits in the DSPI_x_MCR. Debug mode is a mode that the entire device can enter in parallel with the DSPI being configured in one of its module-specific modes.

24.8.1.1 Master mode

In master mode the DSPI can initiate communications with peripheral devices. The DSPI operates as bus master when the MSTR bit in the DSPI_x_MCR is set. The serial communications clock (SCK) is controlled by the master DSPI. All three DSPI configurations are valid in master mode.

In SPI configuration, master mode transfer attributes are controlled by the SPI command in the current TX FIFO entry. The CTAS field in the SPI command selects which of the eight DSPI_x_CTARs set the transfer attributes. Transfer attribute control is on a frame by frame basis.

Refer to [Section 24.8.3, “Serial Peripheral Interface \(SPI\) configuration”](#), for more details.

24.8.1.2 Slave mode

In slave mode the DSPI responds to transfers initiated by an SPI master. The DSPI operates as bus slave when the MSTR bit in the DSPIx_MCR is negated. The DSPI slave is selected by a bus master by having the slave's $\overline{CS0}_x$ asserted. In slave mode, the SCK is provided by the bus master. All transfer attributes are controlled by the bus master, except the clock polarity, clock phase, and the number of bits to transfer. These must be configured in the DSPI slave for correct communications.

24.8.1.3 Module disable mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in module disable mode. The DSPI enters the module disable mode when the MDIS bit in DSPIx_MCR is set.

Refer to [Section 24.8.8, “Power saving features”](#), for more details on the module disable mode.

24.8.1.4 Debug mode

The debug mode is used for system development and debugging. If the MCU enters debug mode while the FRZ bit in the DSPIx_MCR is set, the DSPI stops all serial transfers and enters a stopped state. If the MCU enters debug mode while the FRZ bit is cleared, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI. The DSPI enters debug mode when a debug request is asserted by an external controller.

Refer to [Figure 268](#) for a state diagram.

24.8.2 Start and stop of DSPI transfers

The DSPI has two operating states: STOPPED and RUNNING. The states are independent of DSPI configuration. The default state of the DSPI is STOPPED. In the STOPPED state no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. The TXRXS bit in the DSPIx_SR is cleared in this state. In the RUNNING state, serial transfers take place. The TXRXS bit in the DSPIx_SR is set in the RUNNING state.

[Figure 268](#) shows a state diagram of the start and stop mechanism.

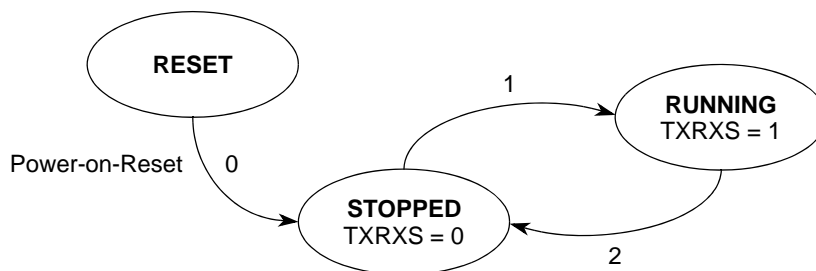


Figure 268. DSPI start and stop state diagram

The transitions are described in [Table 300](#).

Table 300. State transitions for start and stop of DSPI transfers

Transition #	Current State	Next State	Description
0	RESET	STOPPED	Generic power-on-reset transition
1	STOPPED	RUNNING	The DSPI starts (transitions from STOPPED to RUNNING) when all of the following conditions are true: <ul style="list-style-type: none"> • EOQF bit is clear • Debug mode is unselected or the FRZ bit is clear • HALT bit is clear
2	RUNNING	STOPPED	The DSPI stops (transitions from RUNNING to STOPPED) after the current frame for any one of the following conditions: <ul style="list-style-type: none"> • EOQF bit is set • Debug mode is selected and the FRZ bit is set • HALT bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or on the next system clock cycle if no transfers are in progress.

24.8.3 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI configuration when the DCONF field in the DSPIx_MCR is 0b00. The SPI frames can be from 4 to 16 bits long. The data to be transmitted can come from queues stored in RAM external to the DSPI. Host software or an eDMA controller can transfer the SPI data from the queues to a first-in first-out (FIFO) buffer. The received data is stored in entries in the receive FIFO (RX FIFO) buffer. Host software or an eDMA controller transfers the received data from the RX FIFO to memory external to the DSPI.

The FIFO buffer operations are described in [Section 24.8.3.4, “Transmit First In First Out \(TX FIFO\) buffering mechanism”](#), and [Section 24.8.3.5, “Receive First In First Out \(RX FIFO\) buffering mechanism”](#).

The interrupt and DMA request conditions are described in [Section 24.8.7, “Interrupts/DMA requests”](#).

The SPI configuration supports two module-specific modes; master mode and slave mode. The FIFO operations are similar for the master mode and slave mode. The main difference is that in master mode the DSPI initiates and controls the transfer according to the fields in the SPI command field of the TX FIFO entry. In slave mode the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field of the TX FIFO entry is ignored.

24.8.3.1 SPI master mode

In SPI master mode the DSPI initiates the serial transfers by controlling the serial communications clock (\overline{SCK}_x) and the peripheral chip select (\overline{CS}_x) signals. The SPI command field in the executing TX FIFO entry determines which CTARs set the transfer attributes and which \overline{CS}_x signals to assert. The command field also contains various bits that help with queue management and transfer protocol. The data field in the executing TX FIFO entry is loaded into the shift register and shifted out on the serial out (\overline{SOUT}_x)

pin. In SPI master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis.

Refer to [Section 24.7.2.7, “DSPI PUSH TX FIFO Register \(DSPIx_PUSHR\)”](#), for details on the SPI command fields.

24.8.3.2 SPI slave mode

In SPI slave mode the DSPI responds to transfers initiated by an SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase and frame size must be set for successful communication with an SPI master. The SPI slave mode transfer attributes are set in the DSPIx_CTAR0.

24.8.3.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The TX and RX FIFOs are disabled separately. The TX FIFO is disabled by writing a 1 to the DIS_TXF bit in the DSPIx_MCR. The RX FIFO is disabled by writing a 1 to the DIS_RXF bit in the DSPIx_MCR.

The FIFO disable mechanisms are transparent to the user and to host software; transmit data and commands are written to the DSPIx_PUSHR and received data is read from the DSPIx_POPR. When the TX FIFO is disabled, the TFFF, TFUF, and TXCTR fields in DSPIx_SR behave as if there is a one-entry FIFO but the contents of the DSPIx_TXFRs and TXNXTPTR are undefined. When the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the DSPIx_SR behave as if there is a one-entry FIFO but the contents of the DSPIx_RXFRs and POPNXTPTR are undefined.

Disable the TX and RX FIFOs only if the FIFO must be disabled as a requirement of the application's operating mode. A FIFO must be disabled before it is accessed. Failure to disable a FIFO prior to a first FIFO access is not supported, and can result in incorrect results.

24.8.3.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data and SPI commands for transmission. The TX FIFO holds five entries, each consisting of a command field and a data field. SPI commands and data are added to the TX FIFO by writing to the DSPI push TX FIFO register (DSPIx_PUSHR). For more information on DSPIx_PUSHR refer to [Section 24.7.2.7, “DSPI PUSH TX FIFO Register \(DSPIx_PUSHR\)”](#). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO counter field (TXCTR) in the DSPI status register (DSPIx_SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time the DSPI_PUSHR is written or SPI data is transferred into the shift register from the TX FIFO.

Refer to [Section 24.7.2.5, “DSPI Status Register \(DSPIx_SR\)”](#), for more information on DSPIx_SR.

The TXNXTPTR field indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR contains the positive offset from DSPIx_TXFR0 in number of 32-bit registers. For example, TXNXTPTR equal to two means that the DSPIx_TXFR2 contains the SPI data and command for the next

transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register.

24.8.3.4.1 Filling the TX FIFO

Host software or the eDMA controller can add (push) entries to the TX FIFO by writing to the DSPIx_PUSHR. When the TX FIFO is not full, the TX FIFO fill flag (TFFF) in the DSPIx_SR is set. The TFFF bit is cleared when the TX FIFO is full and the eDMA controller indicates that a write to DSPIx_PUSHR is complete or alternatively by host software writing a 1 to the TFFF in the DSPIx_SR. The TFFF can generate a DMA request or an interrupt request.

Refer to [Section 24.8.7.2, “Transmit FIFO fill interrupt or DMA request \(TFFF\)”](#), for details.

The DSPI ignores attempts to push data to a full TX FIFO; that is, the state of the TX FIFO is unchanged. No error condition is indicated.

24.8.3.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO counter is decremented by one. At the end of a transfer, the TCF bit in the DSPIx_SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a 1 to the CLR_TXF bit in DSPIx_MCR.

If an external SPI bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the transmit FIFO underflow flag (TFUF) in the slave's DSPIx_SR is set.

Refer to [Section 24.8.7.4, “Transmit FIFO underflow interrupt request \(TFUF\)”](#), for details.

24.8.3.5 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds five received SPI data frames. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data is removed (popped) from the RX FIFO by reading the DSPIx_POPR register. RX FIFO entries can only be removed from the RX FIFO by reading the DSPIx_POPR or by flushing the RX FIFO.

Refer to [Section 24.7.2.8, “DSPI POP RX FIFO Register \(DSPIx_POPR\)”](#), for more information on the DSPIx_POPR.

The RX FIFO counter field (RXCTR) in the DSPI status register (DSPIx_SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the DSPIx_POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the DSPIx_SR points to the RX FIFO entry that is returned when the DSPIx_POPR is read. The POPNXTPTR contains the positive, 32-bit word offset from DSPIx_RXFR0. For example, POPNXTPTR equal to two means that the DSPIx_RXFR2 contains the received SPI data that is returned when DSPIx_POPR is read. The POPNXTPTR field is incremented every time the DSPIx_POPR is read. POPNXTPTR rolls over every four frames on the MCU.

24.8.3.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the DSPLx_SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the DSPLx_MCR, the data from the transfer that generated the overflow is ignored or put in the shift register. If the ROOE bit is set, the incoming data is put in the shift register. If the ROOE bit is cleared, the incoming data is ignored.

24.8.3.5.2 Draining the RX FIFO

Host software or the eDMA can remove (pop) entries from the RX FIFO by reading the DSPLx_POPR. A read of the DSPLx_POPR decrements the RX FIFO counter by one. Attempts to pop data from an empty RX FIFO are ignored, the RX FIFO counter remains unchanged. The data returned from reading an empty RX FIFO is undetermined.

Refer to [Section 24.7.2.8, “DSPI POP RX FIFO Register \(DSPLx_POPR\)”](#), for more information on DSPLx_POPR.

When the RX FIFO is not empty, the RX FIFO drain flag (RFDF) in the DSPLx_SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the eDMA controller indicates that a read from DSPLx_POPR is complete; alternatively the RFDF bit can be cleared by the host writing a 1 to it.

24.8.4 DSPI baud rate and clock delay generation

The SCK_x frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option of doubling the baud rate.

[Figure 269](#) shows conceptually how the SCK signal is generated.

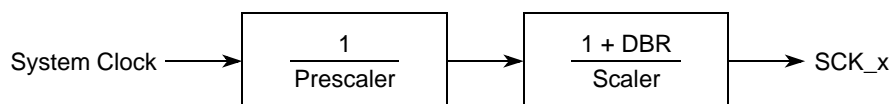


Figure 269. Communications clock prescalers and scalers

24.8.4.1 Baud rate generator

The baud rate is the frequency of the serial communication clock (SCK_x). The system clock is divided by a baud rate prescaler (defined by DSPLx_CTAR[PBR]) and baud rate scaler (defined by DSPLx_CTAR[BR]) to produce SCK_x with the possibility of doubling the baud rate. The DBR, PBR, and BR fields in the DSPLx_CTARs select the frequency of SCK_x using the following formula:

Eqn. 5

$$\text{SCK baud rate} = \frac{f_{\text{SYS}}}{\text{PBRPrescalerValue}} \times \frac{1 + \text{DBR}}{\text{BRScalerValue}}$$

Table 301 shows an example of a computed baud rate.

Table 301. Baud rate computation example

f_{SYS}	PBR	Prescaler value	BR	Scaler value	DBR value	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mbit/s
20 MHz	0b00	2	0b0000	2	1	10 Mbit/s

24.8.4.2 CS to SCK delay (t_{CSC})

The $\overline{\text{CS}}_x$ to SCK $_x$ delay is the length of time from assertion of the $\overline{\text{CS}}_x$ signal to the first SCK $_x$ edge. Refer to Figure 271 for an illustration of the $\overline{\text{CS}}_x$ to SCK $_x$ delay. The PCSSCK and CSSCK fields in the DSPI $_x$ _CTAR n registers select the $\overline{\text{CS}}_x$ to SCK $_x$ delay, and the relationship is expressed by the following formula:

$$t_{\text{CSC}} = \frac{1}{f_{\text{SYS}}} \times \text{PCSSCK} \times \text{CSSCK}$$

Eqn. 6

Table 302 shows an example of the computed $\overline{\text{CS}}$ to SCK $_x$ delay.

Table 302. $\overline{\text{CS}}$ to SCK delay computation example

PCSSCK	Prescaler value	CSSCK	Scaler value	f_{SYS}	$\overline{\text{CS}}$ to SCK delay
0b01	3	0b0100	32	100 MHz	0.96 μs

24.8.4.3 After SCK delay (t_{ASC})

The after SCK $_x$ delay is the length of time between the last edge of SCK $_x$ and the deassertion of $\overline{\text{CS}}_x$. Refer to Figure 271 and Figure 272 for illustrations of the after SCK $_x$ delay. The PASC and ASC fields in the DSPI $_x$ _CTAR n registers select the after SCK delay. The relationship between these variables is given in the following formula:

$$t_{\text{ASC}} = \frac{1}{f_{\text{SYS}}} \times \text{PASC} \times \text{ASC}$$

Eqn. 7

Table 303 shows an example of the computed after SCK delay.

Table 303. After SCK delay computation example

PASC	Prescaler value	ASC	Scaler value	f_{SYS}	After SCK delay
0b01	3	0b0100	32	100 MHz	0.96 μs

24.8.4.4 Delay after transfer (t_{DT})

The delay after transfer is the length of time between negation of the \overline{CSx} signal for a frame and the assertion of the \overline{CSx} signal for the next frame. The PDT and DT fields in the DSPLx_CTARn registers select the delay after transfer.

Refer to [Figure 271](#) for an illustration of the delay after transfer.

The following formula expresses the PDT/DT/delay after transfer relationship:

Eqn. 8

$$t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$$

[Table 304](#) shows an example of the computed delay after transfer.

Table 304. Delay after transfer computation example

PDT	Prescaler value	DT	Scaler value	f_{SYS}	Delay after transfer
0b01	3	0b1110	32768	100 MHz	0.98 ms

24.8.4.5 Peripheral Chip Select strobe enable ($\overline{CS5_x}$)

The $\overline{CS5_x}$ signal provides a delay to allow the \overline{CSx} signals to settle after transitioning, thereby avoiding glitches. When the DSPI is in master mode and PCSSE bit is set in the DSPLx_MCR, $\overline{CS5_x}$ provides a signal for an external demultiplexer to decode the $\overline{CS4_x}$ signals into as many as 32 glitch-free \overline{CSx} signals.

[Figure 270](#) shows the timing of the $\overline{CS5_x}$ signal relative to \overline{CS} signals.

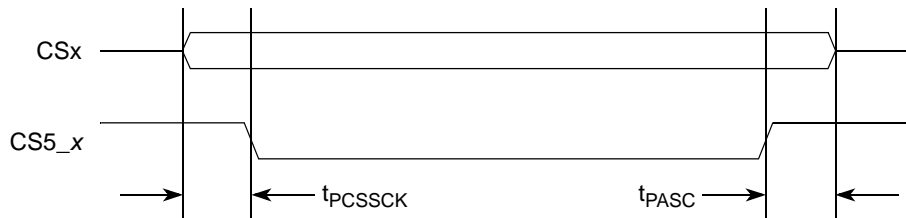


Figure 270. Peripheral Chip Select strobe timing

The delay between the assertion of the \overline{CSx} signals and the assertion of $\overline{CS5_x}$ signal is selected by the PCSSCK field in the DSPLx_CTAR based on the following formula:

Eqn. 9

$$t_{PCSSCK} = \frac{1}{f_{SYS}} \times PCSSCK$$

At the end of the transfer the delay between $\overline{CS5_x}$ negation and \overline{CSx} negation is selected by the PASC field in the DSPLx_CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{SYS}}} \times \text{PASC}$$

Table 305 shows an example of the computed t_{PCSSCK} delay.

Table 305. Peripheral Chip Select strobe assert computation example

PCSSCK	Prescaler	f_{SYS}	Delay before transfer
0b11	7	100 MHz	70.0 ns

Table 306 shows an example of the computed the t_{PASC} delay.

Table 306. Peripheral Chip Select strobe negate computation example

PASC	Prescaler	f_{SYS}	Delay after transfer
0b11	7	100 MHz	70.0 ns

24.8.5 Transfer formats

The SPI serial communication is controlled by the serial communications clock (SCK_x) signal and the CS_x signals. The SCK_x signal provided by the master device synchronizes shifting and sampling of the data by the SIN_x and SOUT_x pins. The CS_x signals serve as enable signals for the slave devices.

When the DSPI is the bus master, the CPOL and CPHA bits in the DSPI clock and transfer attributes registers (DSPI_x_CTAR_n) select the polarity and phase of the serial clock, SCK_x. The polarity bit selects the idle state of the SCK_x. The clock phase bit selects if the data on SOUT_x is valid before or on the first SCK_x edge.

When the DSPI is the bus slave, CPOL and CPHA bits in the DSPI_x_CTAR0 (SPI slave mode) select the polarity and phase of the serial clock. Even though the bus slave does not control the SCK signal, clock polarity, clock phase and number of bits to transfer must be identical for the master device and the slave device to ensure proper transmission.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0
- Classic SPI with CPHA = 1
- Modified transfer format with CPHA = 0
- Modified transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the DSPI_x_MCR selects between classic SPI format and modified transfer format. The classic SPI formats are described in Section 24.8.5.1, “Classic SPI transfer format (CPHA = 0)”, and Section 24.8.5.2, “Classic SPI transfer format (CPHA = 1)”. The modified transfer formats are described in Section 24.8.5.3, “Modified SPI transfer format (MTFE = 1, CPHA = 0)”, and Section 24.8.5.4, “Modified SPI transfer format (MTFE = 1, CPHA = 1)”.

In the SPI configuration, the DSPI provides the option of keeping the CS signals asserted between frames. Refer to [Section 24.8.5.5, “Continuous selection format”](#), for details.

24.8.5.1 Classic SPI transfer format (CPHA = 0)

The transfer format shown in [Figure 271](#) is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN_x pins on the odd-numbered SCK_x edges and change the data on their SOUT_x pins on the even-numbered SCK_x edges.

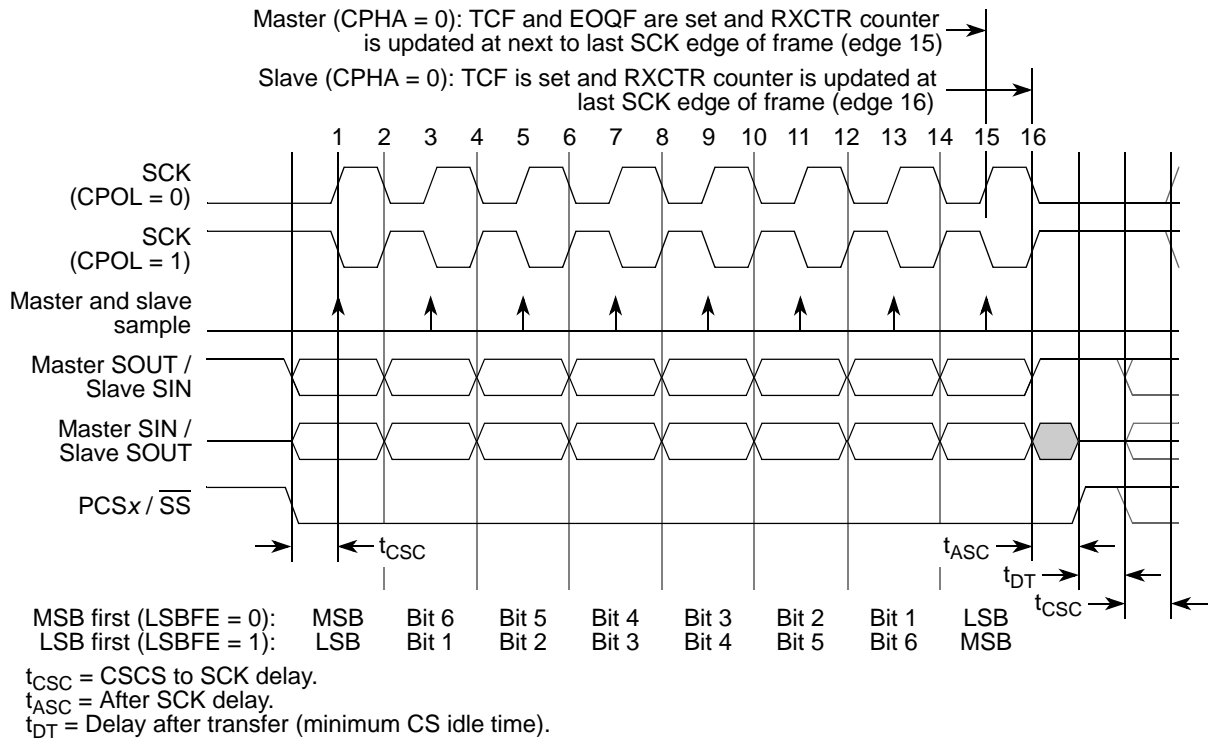


Figure 271. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

The master initiates the transfer by placing its first data bit on the SOUT_x pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT_x pin. After the t_{CSC} delay has elapsed, the master outputs the first edge of SCK_x. This is the edge used by the master and slave devices to sample the first input data bit on their serial data input signals. At the second edge of the SCK_x the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN_x pins on the odd-numbered clock edges and changes the data on their SOUT_x pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the CS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

For the CPHA = 0 condition of the master, TCF and EOQF are set and the RXCTR counter is updated at the next to last serial clock edge of the frame (edge 15) of [Figure 271](#).

For the CPHA = 0 condition of the slave, TCF is set and the RXCTR counter is updated at the last serial clock edge of the frame (edge 16) of [Figure 271](#).

24.8.5.2 Classic SPI transfer format (CPHA = 1)

The transfer format shown in Figure 272 is used to communicate with peripheral SPI slave devices that require the first SCK_x edge before the first data bit becomes available on the slave SOUT_x pin. In this format the master and slave devices change the data on their SOUT_x pins on the odd-numbered SCK_x edges and sample the data on their SIN_x pins on the even-numbered SCK_x edges.

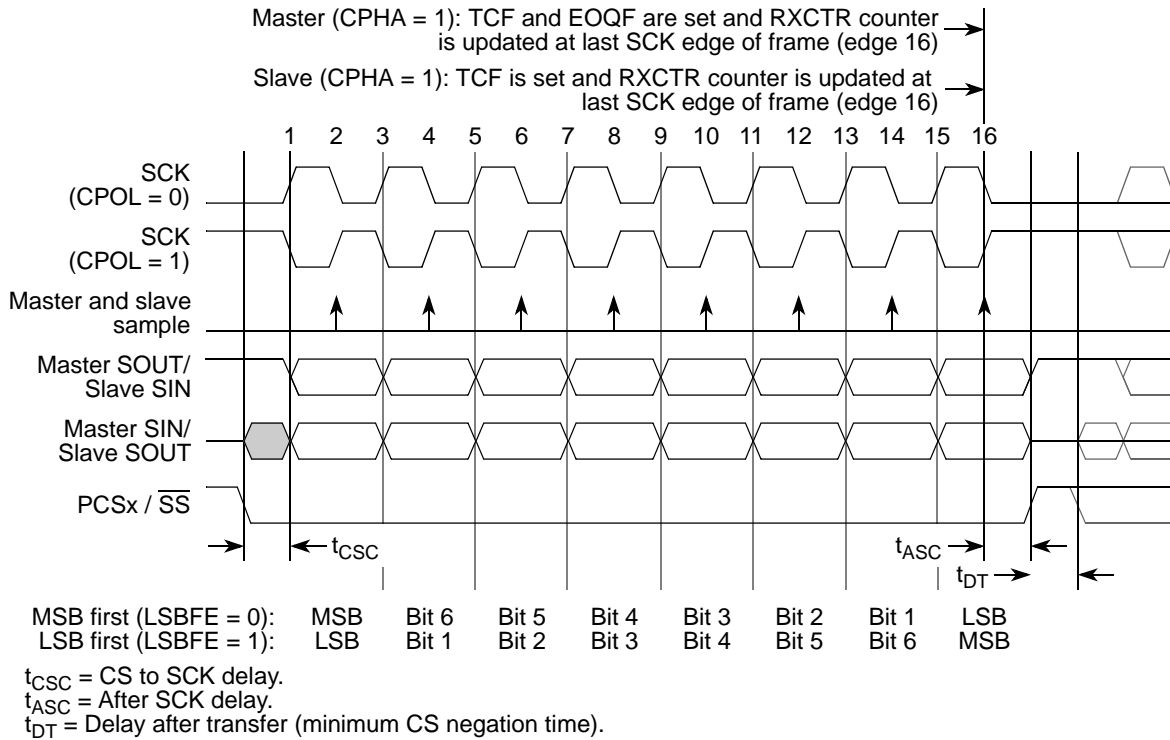


Figure 272. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)

The master initiates the transfer by asserting the CS_x signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK_x edge and at the same time places valid data on the master SOUT_x pin. The slave responds to the first SCK_x edge by placing its first data bit on its slave SOUT_x pin.

At the second edge of the SCK_x the master and slave sample their SIN_x pins. For the rest of the frame the master and the slave change the data on their SOUT_x pins on the odd-numbered clock edges and sample their SIN_x pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the CS_x signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

For CPHA = 1 the master EOQF and TCF and slave TCF are set at the last serial clock edge (edge 16) of Figure 272. For CPHA = 1 the master and slave RXCTR counters are updated on the same clock edge.

24.8.5.3 Modified SPI transfer format (MTFE = 1, CPHA = 0)

In this modified transfer format both the master and the slave sample later in the SCK period than in classic SPI mode to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

NOTE

For the modified transfer format to operate correctly, you must thoroughly analyze the SPI link timing budget.

The master and the slave place data on the SOUT_x pins at the assertion of the CS_x signal. After the CS_x to SCK_x delay has elapsed the first SCK_x edge is generated. The slave samples the master SOUT_x signal on every odd numbered SCK_x edge. The slave also places new data on the slave SOUT_x on every odd numbered clock edge.

The master places its second data bit on the SOUT_x line one system clock after odd numbered SCK_x edge. The point where the master samples the slave SOUT_x is selected by writing to the SMPL_PT field in the DSPL_x_MCR. [Table 307](#) lists the number of system clock cycles between the active edge of SCK_x and the master sample point for different values of the SMPL_PT bit field. The master sample point can be delayed by one or two system clock cycles.

Table 307. Delayed master sample point

SMPL_PT	Number of system clock cycles between odd-numbered edge of SCK and sampling of SIN
00	0
01	1
10	2
11	Invalid value

[Figure 273](#) shows the modified transfer format for CPHA = 0. Only the condition where CPOL = 0 is illustrated. The delayed master sample points are indicated with a lighter shaded arrow.

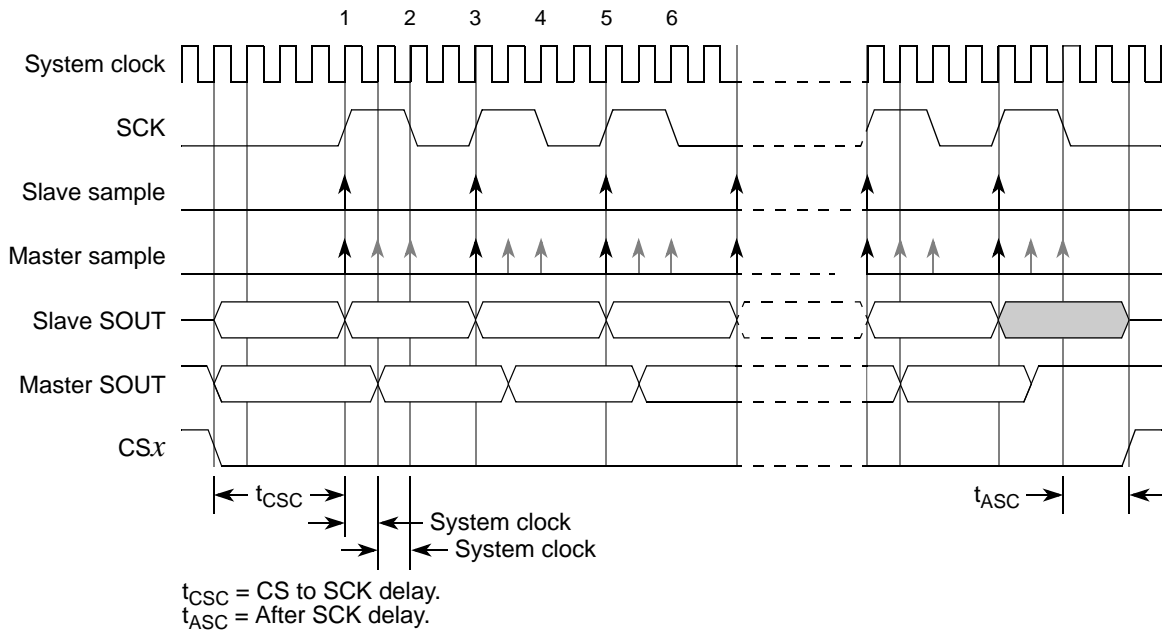


Figure 273. DSPI modified transfer format (MTEF = 1, CPHA = 0, $f_{SCK} = f_{SYS} / 4$)

24.8.5.4 Modified SPI transfer format (MTFE = 1, CPHA = 1)

At the start of a transfer the DSPI asserts the CS signal to the slave device. After the CS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the 3rd SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge is visible on the master SCK pin during the sampling of the last bit. The SCK to CS delay must be programmed to be greater than or equal to half the SCK period.

NOTE

For the modified transfer format to operate correctly, you must thoroughly analyze the SPI link timing budget.

Figure 274 shows the modified transfer format for CPHA = 1. Only the condition where CPOL = 0 is shown.

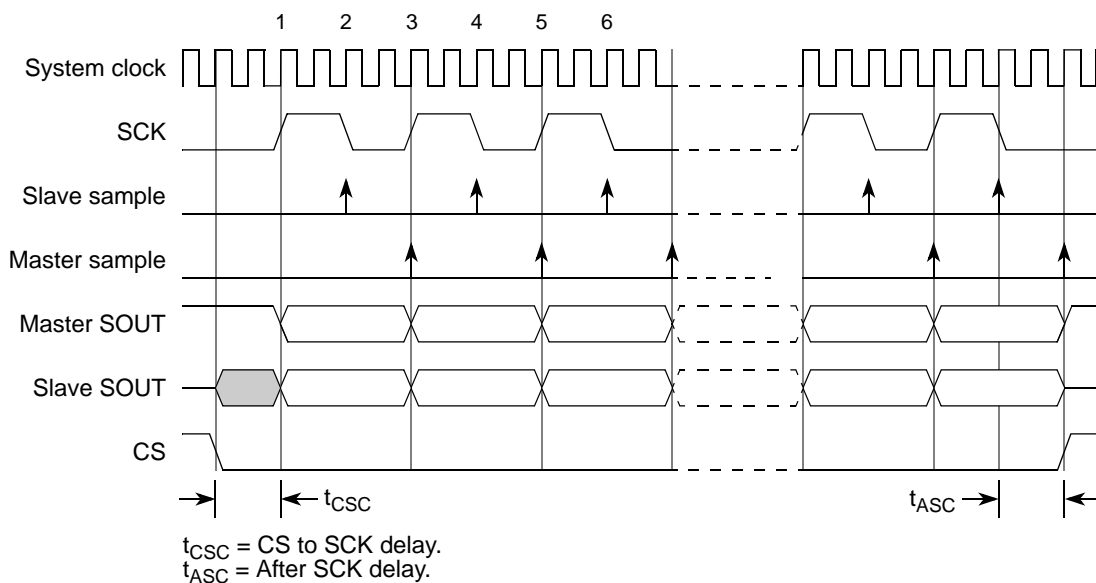


Figure 274. DSPI modified transfer format (MTFE = 1, CPHA = 1, $f_{SCK} = f_{SYS} / 4$)

24.8.5.5 Continuous selection format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The continuous selection format provides the flexibility to handle both cases. The continuous selection format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the DSPI drives the asserted chip select signals to their idle states in between frames. The idle states of the chip select signals are selected by the PCSIS field in the DSPIx_MCR.

Figure 275 shows the timing diagram for two 4-bit transfers with CPHA = 1 and CONT = 0.

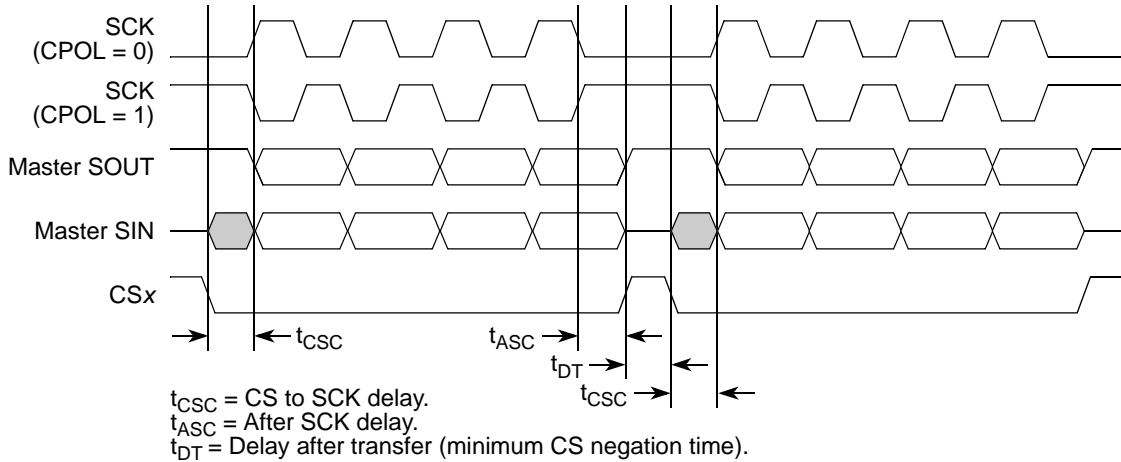


Figure 275. Example of non-continuous format (CPHA = 1, CONT = 0)

When the CONT = 1 and the CS signal for the next transfer is the same as for the current transfer, the CS signal remains asserted for the duration of the two transfers. The delay between transfers (t_{DT}) is not inserted between the transfers.

Figure 276 shows the timing diagram for two 4-bit transfers with CPHA = 1 and CONT = 1.

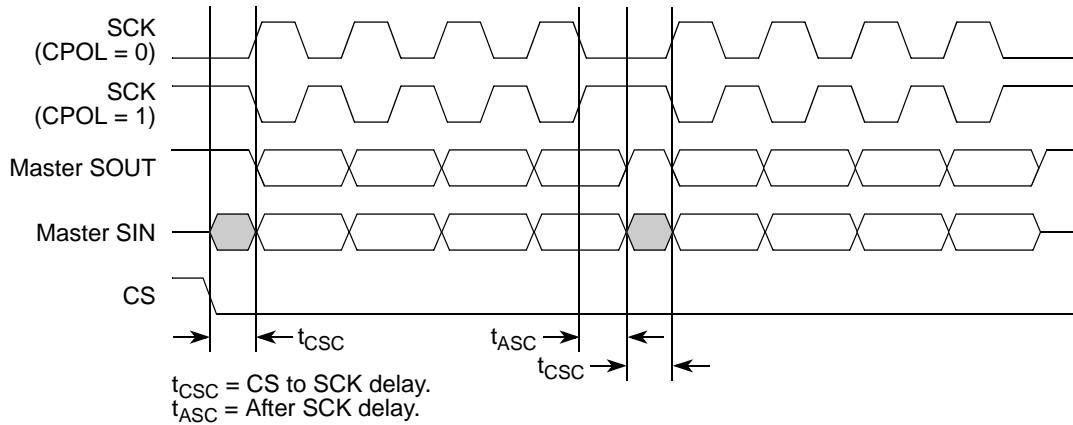


Figure 276. Example of continuous transfer (CPHA = 1, CONT = 1)

In Figure 276, the period length at the start of the next transfer is the sum of t_{ASC} and t_{CSC} ; i.e., it does not include a half-clock period. The default settings for these provide a total of four system clocks. In many situations, t_{ASC} and t_{CSC} must be increased if a full half-clock period is required.

Switching CTARs between frames while using continuous selection can cause errors in the transfer. The CS signal must be negated before CTAR is switched.

When the CONT bit = 1 and the CS signals for the next transfer are different from the present transfer, the CS signals behave as if the CONT bit was not set.

NOTE

User must fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty. For example; while transmitting in master mode, ensure that the last entry in the TXFIFO, after which TXFIFO becomes empty, has $CONT = 0$ in the command frame.

When operating in slave mode, ensure that when the last-entry in the TXFIFO is completely transmitted (i.e. the corresponding TCF flag is asserted and TXFIFO is empty) the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

24.8.5.6 Clock polarity switching between DSPI transfers

If it is desired to switch polarity between non-continuous DSPI frames, the edge generated by the change in the idle state of the clock occurs one system clock before the assertion of the chip select for the next frame.

Refer to [Section 24.7.2.4, “DSPI Clock and Transfer Attributes Registers 0–7 \(DSPIx_CTARn\)”](#).

In [Figure 277](#), time ‘A’ shows the one clock interval. Time ‘B’ is user programmable from a minimum of two system clocks.

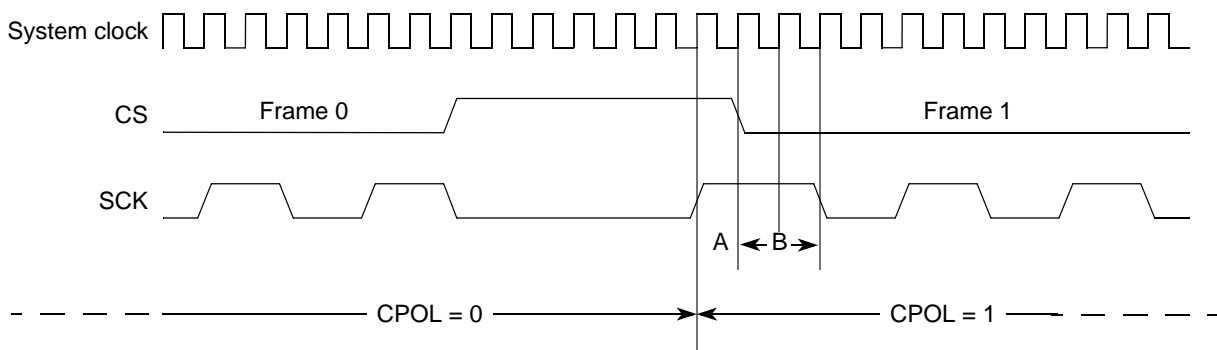


Figure 277. Polarity switching between frames

24.8.5.7 Fast Continuous Selection Format

The Fast Continuous Selection Format is same as the Continuous Selection Format except that the inter command delays i.e. t_{ASC} and t_{CSC} can be masked out and are not inserted by the hardware.

NOTE

The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

The Fast Continuous Selection Format is enabled by writing ‘1’ into bit 2 of the DSPI_MCR register. When this bit is asserted, the bits 25:24 of the DSPI_PUSHR register perform the function of mask bits

for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and speed of the transfer process, the software can simply mask these delays while programming the command in the DSPI_PUSHR register.

While masking the delays the software must follow the following masking rules, else, correct operation is not guaranteed.

- Mask t_{ASC} bit masks the “After SCK” delay for the current frame
- Mask t_{CSC} bit masks the “PCS to SCK” delay for the next frame
- “After SCK” or ASC delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and DSPI_MCR[2] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here too.

Figure 278 shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both ASC & CSC delays masked except for the last frame that terminated the transfer. The last frame has ASC delay at its end.

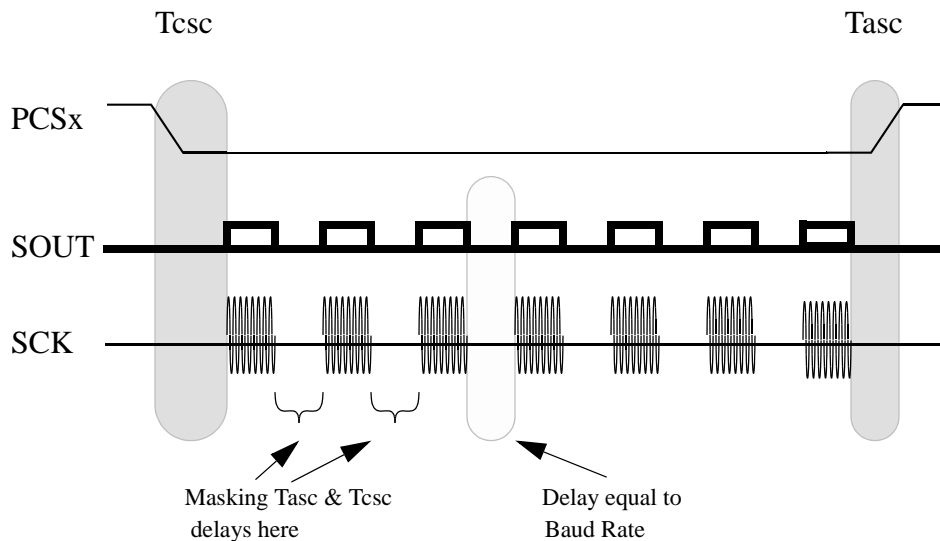


Figure 278. Example of Fast Continuous Selection Format

In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

NOTE

All frames need to be masked when operating in Fast Continuous Mode. User cannot switch to continuous mode (delays not masked) and return to the fast continuous mode when Fast Continuous Mode of operation is selected.

CPHA is to be kept "1" when operating in Fast Continuous Mode.

24.8.6 Continuous Serial communications clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the DSPIx_MCR. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Setting CPHA = 0 is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for modified transfer format.

Clock and transfer attributes for the continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the continuous SCK mode is terminated.

The device is designed to use the same baud rate for all transfers made while using the continuous SCK. Switching clock polarity between frames while using continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into module disable mode.

Enabling continuous SCK disables the CS to SCK delay and the After SCK delay. The delay after transfer is fixed at one SCK cycle. [Figure 279](#) shows timing diagram for continuous SCK format with continuous selection disabled.

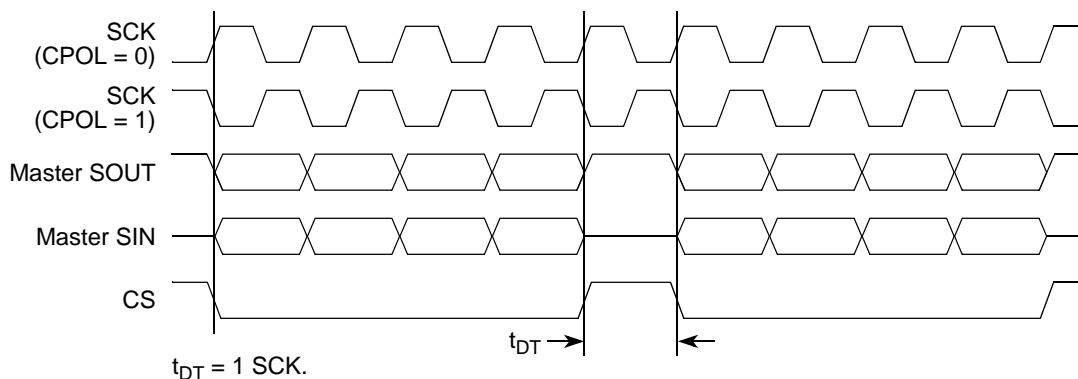


Figure 279. Continuous SCK timing diagram (CONT = 0)

If the CONT bit in the TX FIFO entry is set, CS remains asserted between the transfers when the CS signal for the next transfer is the same as for the current transfer. [Figure 280](#) shows timing diagram for continuous SCK format with continuous selection enabled.

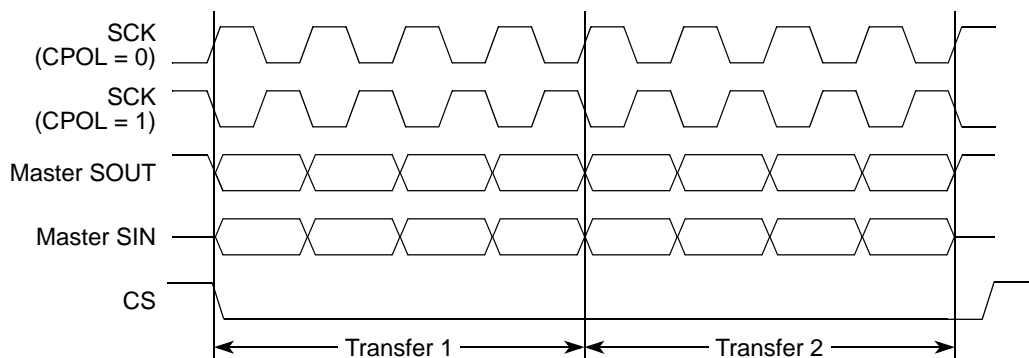


Figure 280. Continuous SCK timing diagram (CONT = 1)

24.8.7 Interrupts/DMA requests

The DSPI has conditions that can generate interrupt requests only, and conditions that can generate interrupts or DMA requests. [Table 308](#) lists these conditions.

Table 308. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of transfer queue has been reached (EOQ)	EOQF	X	
TX FIFO is not full	TFFF	X	X
Current frame transfer is complete	TCF	X	
TX FIFO underflow has occurred	TFUF	X	
RX FIFO is not empty	RFDF	X	X
RX FIFO overflow occurred	RFOF	X	
A FIFO overrun occurred ¹	TFUF ORed with RFOF	X	

¹ The FIFO overrun condition is created by ORing the TFUF and RFOF flags together.

Each condition has a flag bit and a request enable bit. The flag bits are described in the [Section 24.7.2.5, “DSPI Status Register \(DSPIx_SR\)”](#), and the request enable bits are described in the [Section 24.7.2.6, “DSPI DMA / Interrupt Request Select and Enable Register \(DSPIx_RSER\)”](#). The TX FIFO fill flag (TFFF) and RX FIFO drain flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the DSPIx_RSER.

24.8.7.1 End of queue interrupt request (EOQF)

The end of queue request indicates that the end of a transmit queue is reached. The end of queue request is generated when the EOQ bit in the executing SPI command is asserted and the EOQF_RE bit in the DSPIx_RSER is set. Refer to the EOQ bit description in [Section 24.7.2.5, “DSPI Status Register \(DSPIx_SR\)”](#). Refer to [Figure 271](#) and [Figure 272](#) that illustrate when EOQF is set.

24.8.7.2 Transmit FIFO fill interrupt or DMA request (TFFF)

The transmit FIFO fill request indicates that the TX FIFO is not full. The transmit FIFO fill request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the DSPLx_RSER is set. The TFFF_DIRS bit in the DSPLx_RSER selects whether a DMA request or an interrupt request is generated.

24.8.7.3 Transfer complete interrupt request (TCF)

The transfer complete request indicates the end of the transfer of a serial frame. The transfer complete request is generated at the end of each frame transfer when the TCF_RE bit is set in the DSPLx_RSER. Refer to the TCF bit description in [Section 24.7.2.5, “DSPI Status Register \(DSPLx_SR\)”](#). Refer to [Figure 271](#) and [Figure 272](#), which show when TCF is set.

24.8.7.4 Transmit FIFO underflow interrupt request (TFUF)

The transmit FIFO underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in slave mode and SPI configuration is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the DSPLx_RSER is set, an interrupt request is generated.

24.8.7.5 Receive FIFO drain interrupt or DMA request (RFDF)

The receive FIFO drain request indicates that the RX FIFO is not empty. The receive FIFO drain request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the DSPLx_RSER is set. The RFDF_DIRS bit in the DSPLx_RSER selects whether a DMA request or an interrupt request is generated.

24.8.7.6 Receive FIFO overflow interrupt request (RFOF)

The receive FIFO overflow request indicates that an overflow condition in the RX FIFO has occurred. A receive FIFO overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the DSPLx_RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the DSPLx_MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is negated, the incoming data is ignored.

24.8.7.7 FIFO overrun request (TFUF) or (RFOF)

The FIFO overrun request indicates that at least one of the FIFOs in the DSPI has exceeded its capacity. The FIFO overrun request is generated by logically OR'ing together the RX FIFO overflow and TX FIFO underflow signals.

24.8.8 Power saving features

The DSPI supports two power-saving strategies:

- Module disable mode—clock gating of non-memory mapped logic
- Clock gating of slave interface signals and clock to memory-mapped logic

24.8.8.1 Module disable mode

Module disable mode is a module-specific mode that the DSPI can enter to save power. Host software can initiate the module disable mode by writing a 1 to the MDIS bit in the DSPI_x_MCR. In module disable mode, the DSPI is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different affect when the DSPI is in the module disable mode. Reading the RX FIFO pop register does not change the state of the RX FIFO. Likewise, writing to the TX FIFO push register does not change the state of the TX FIFO. Clearing either of the FIFOs does not have any effect in the module disable mode. Changes to the DIS_TXF and DIS_RXF fields of the DSPI_x_MCR does not have any affect in the module disable mode. In the module disable mode, all status bits and register flags in the DSPI return the correct values when read, but writing to them has no affect. Writing to the DSPI_x_TCR during module disable mode does not have an effect. Interrupt and DMA request signals cannot be cleared while in the module disable mode.

24.9 Initialization and application information

24.9.1 Managing queues

DSPI queues are not part of the DSPI module, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI configuration. This section presents an example of how to manage queues for the DSPI.

1. The last command word from a queue is executed. The EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the DSPI_x_SR is set.
3. The setting of the EOQF flag disables both serial transmission, and serial reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is negated to indicate the STOPPED state.
4. The eDMA continues to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the eDMA controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in DSPI_x_SR or by checking RFDF in the DSPI_x_SR after each read operation of the DSPI_x_POPR.
7. Modify DMA descriptor of TX and RX channels for “new” queues.
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the DSPI_x_MCR register and flush the RX FIFO by writing a 1 to the CLR_RXF bit in the DSPI_x_MCR register.

9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the DSPIx_TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

24.9.2 Baud rate settings

Table 309 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the DSPIx_CTARs. The values are calculated at a 100 MHz system frequency.

Table 309. Baud rate values

		Baud Rate divider prescaler values (DSPI_CTAR[PBR])			
		2	3	5	7
Baud rate scaler values (DSPI_CTAR[BR])	2	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
	4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
	6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
	8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
	16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
	32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
	64	781 kHz	521 kHz	312 kHz	223 kHz
	128	391 kHz	260 kHz	156 kHz	112 kHz
	256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
	512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
	1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
	2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
	4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
	8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz	
32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz	

24.9.3 Delay settings

Table 310 shows the values for the delay after transfer (t_{DT}) and CS to SCK delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the DSPIx_CTARs. The values calculated assume a 100 MHz system frequency.

Table 310. Delay values

		Delay prescaler values (DSPI_CTAR[PBR])			
		1	3	5	7
Delay scaler values (DSPI_CTAR[DT])	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 μ s
	32	320.0 ns	960.0 ns	1.6 μ s	2.2 μ s
	64	640.0 ns	1.9 μ s	3.2 μ s	4.5 μ s
	128	1.3 μ s	3.8 μ s	6.4 μ s	9.0 μ s
	256	2.6 μ s	7.7 μ s	12.8 μ s	17.9 μ s
	512	5.1 μ s	15.4 μ s	25.6 μ s	35.8 μ s
	1024	10.2 μ s	30.7 μ s	51.2 μ s	71.7 μ s
	2048	20.5 μ s	61.4 μ s	102.4 μ s	143.4 μ s
	4096	41.0 μ s	122.9 μ s	204.8 μ s	286.7 μ s
	8192	81.9 μ s	245.8 μ s	409.6 μ s	573.4 μ s
	16384	163.8 μ s	491.5 μ s	819.2 μ s	1.1 ms
	32768	327.7 μ s	983.0 μ s	1.6 ms	2.3 ms
65536	655.4 μ s	2.0 ms	3.3 ms	4.6 ms	

24.9.4 MPC5604E DSPI compatibility with QSPI of the MPC500 MCUs

Table 311 shows the translation of commands written to the TX FIFO command halfword with commands written to the command RAM of the MPC500 family MCU QSPI. The table illustrates how to configure the DSPIx_CTARs to match the default cases for the possible combinations of the MPC500 family control bits in its command RAM. The defaults for the QSPI are based on a system clock of 40 MHz.

The following delay variables generate the same delay, or as close as possible, from the DSPI_100 MHz system clock that a QSPI generates from a 40 MHz system clock. For other system clock frequencies, you can recompute the values using the information presented in [Section 24.9.3, “Delay settings”](#).¹

- For BITSE = 0 → 8 bits per transfer.
- For DT = 0 → 0.425 μs delay: for this value, the closest value in the DSPI is 0.480 μs.
- For DSCK = 0 → 0.5 of the SCK period: for this value, the value for the DSPI is 20 ns.

Table 311. MPC5604E QSPI compatibility with the DSPI

MPC5604E family control bits DSPI corresponding control bits						Corresponding DSPIx_CTAR register configuration					
BITSE	CTAS[0]	DT	CTAS[1]	DSCK	CTAS[2]	DSPIx_CTARx	FMSZ	PDT	DT	PCSSCK	CSSCK
0		0		0		0	0111	10	0011	00	0000
0		0		1		1	0111	10	0011	User	User
0		1		0		2	0111	User ¹	User	00	0000
0		1		1		3	0111	User	User	User	User
1		0		0		4	User	10	0011	00	0000
1		0		1		5	User	10	0011	User	User
1		1		0		6	User	User	User	00	0000
1		1		1		7	User	User	User	User	User

¹ Selected by user.

24.9.5 Calculation of FIFO pointer addresses

The user has complete visibility of the TX and RX FIFO contents through the FIFO registers, and valid entries can be identified through a memory mapped pointer and a memory mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the transmit next pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the pop next pointer (POPNXTPTR).

Refer to [Section 24.8.3.4, “Transmit First In First Out \(TX FIFO\) buffering mechanism”](#), and [Section 24.8.3.5, “Receive First In First Out \(RX FIFO\) buffering mechanism”](#), for details on the FIFO operation. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO.

[Figure 281](#) illustrates the concept of first-in and last-in FIFO entries along with the FIFO counter.

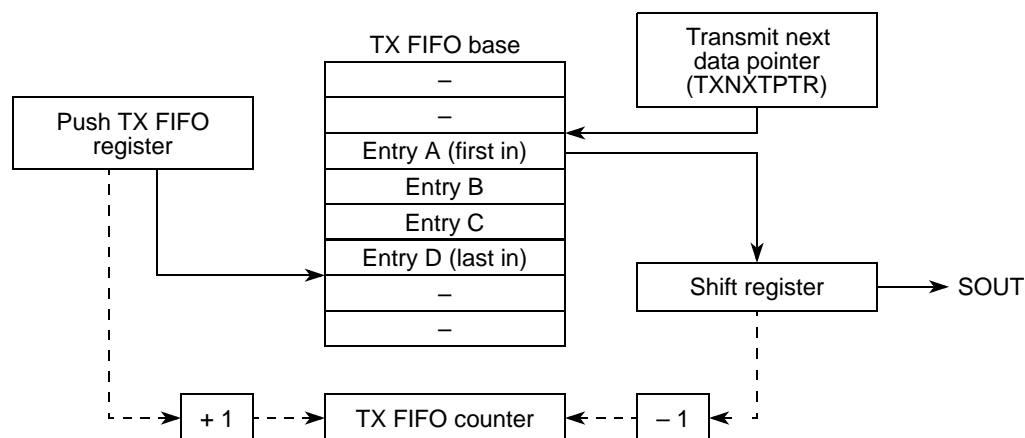


Figure 281. TX FIFO pointers and counter

24.9.5.1 Address calculation for first-in entry and last-in entry in TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

Eqn. 11

$$\text{First-in entry address} = \text{TXFIFO base} + 4 (\text{TXNXPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Eqn. 12

$$\text{Last-in entry address} = \text{TXFIFO base} + 4 \times [(\text{TXCTR} + \text{TXNXPTR} - 1) \text{ modulo TXFIFO depth}]$$

where:

- TXFIFO base = base address of transmit FIFO
- TXCTR = transmit FIFO counter
- TXNXPTR = transmit next pointer
- TX FIFO depth = transmit FIFO depth (depth is 5)

24.9.5.2 Address calculation for first-in entry and last-in entry in RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

Eqn. 13

$$\text{First-in entry address} = \text{RXFIFO base} + 4 \times (\text{POPXPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Eqn. 14

$$\text{Last-in entry address} = \text{RXFIFO base} + 4 \times [(\text{RXCTR} + \text{POPNXPTR} - 1) \text{ modulo RXFIFO depth}]$$

where:

RXFIFO base = base address of receive FIFO

RXCTR = receive FIFO counter

POPNXPTR = pop next pointer

RX FIFO depth = receive FIFO depth (depth is 5)

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 25

LIN Controller (LINFlex)

25.1 Introduction

The LINFlex controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode LINFlex autonomously handles the LIN messages once software has triggered the header transmission until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

25.2 Main features

The LINFlex on the device features the following:

- Supports LIN Master mode, LIN Slave mode and UART mode
- LIN state machine compliant to LIN1.3, 2.0, and 2.1 Specifications
- Handles LIN frame transmission and reception without CPU intervention
- LIN features
 - Autonomous LIN frame handling
 - LIN0 supports master and slave mode with 16 identifier filters
 - LIN1 supports master mode only (no identifier filters required)
 - Message buffer to store Identifier and as much as 8 data bytes
 - Supports message length as long as 64 bytes
 - Detection and flagging of LIN errors: Sync field; Delimiter; ID parity; Bit;
 - Framing; Checksum and Time-out errors
 - Classic or extended checksum calculation
 - Configurable Break duration as long as 36-bit times
 - Programmable Baud rate pre-scalers (13-bit mantissa, 4-bit fractional)
 - Diagnostic features: Loop back; Self Test; LIN bus stuck dominant detection
 - Interrupt-driven operation with 16 interrupt sources
- LIN slave mode features
 - Autonomous LIN header handling
 - Autonomous LIN response handling
- UART mode
 - Full-duplex operation
 - Standard non return-to-zero (NRZ) mark/space format

25.3 General description

The increasing number of communication peripherals embedded on microcontrollers, for example CAN, LIN and SPI, requires more and more CPU resources for communication management. Even a 32-bit microcontroller is overloaded if its peripherals do not provide high-level features to autonomously handle the communication.

Even though the LIN protocol with a maximum baud rate of 20 kbps is relatively slow, it still generates a non-negligible load on the CPU if the LIN is implemented on a standard UART, as usually the case.

To minimize the CPU load in Master mode, LINFlex handles the LIN messages autonomously.

In Master mode, once the software has triggered the header transmission, LINFlex does not request any software intervention until the next header transmission request in transmission mode or until the checksum reception in reception mode.

To minimize the CPU load in Slave mode, LINFlex requires software intervention only to:

- Trigger transmission or reception or data discard depending on the identifier
- Write data into the buffer (transmission mode) or read data from the buffer (reception mode) after checksum reception

If filter mode is activated for Slave mode, LINFlex requires software intervention only to write data into the buffer (transmission mode) or read data from the buffer (reception mode)

The software uses the control, status and configuration registers to:

- Configure LIN parameters (for example, baud rate or mode)
- Request transmissions
- Handle receptions
- Manage interrupts
- Configure LIN error and timeout detection
- Process diagnostic information

The message buffer stores transmitted or received LIN frames.

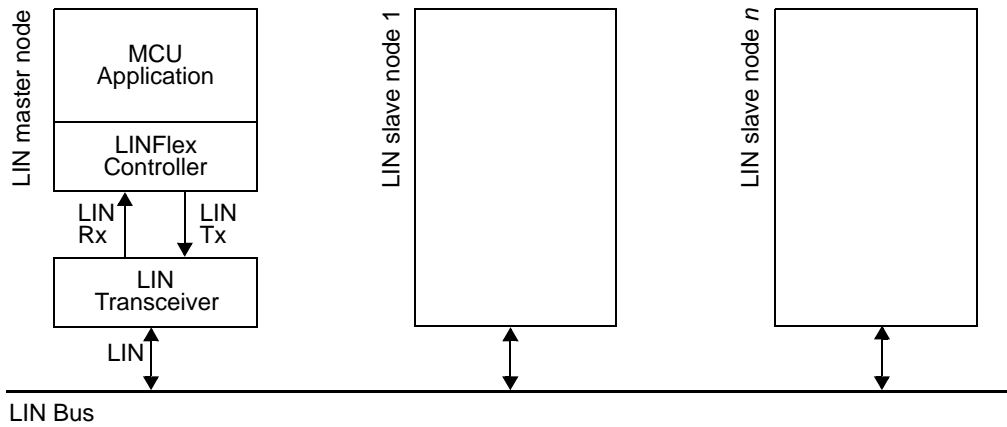
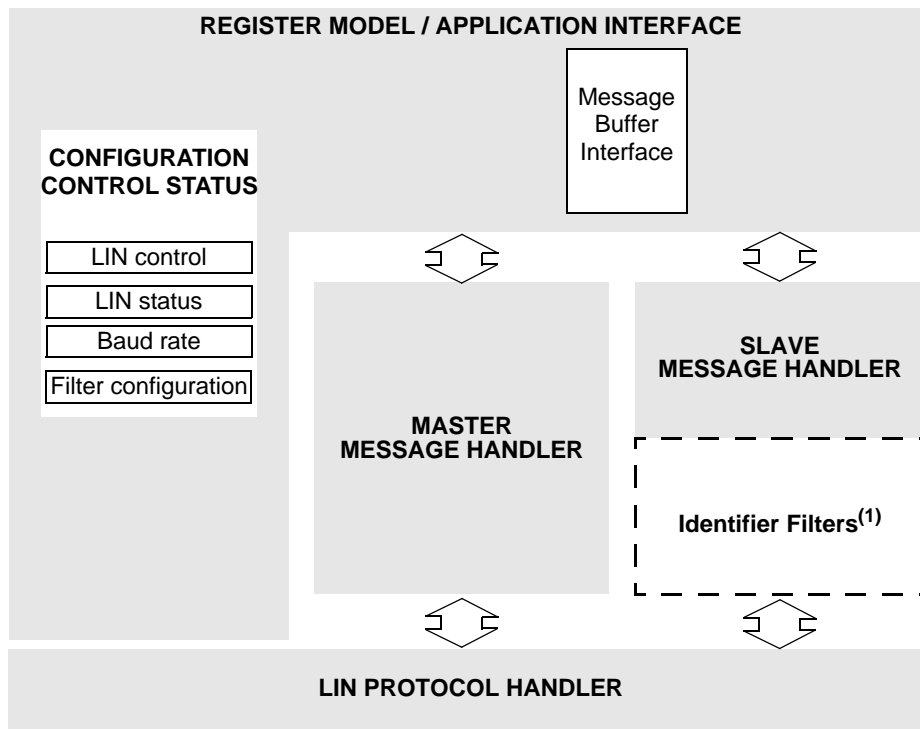


Figure 282. LIN topology network



1. Filter activation optional

Figure 283. LINFlex block diagram

25.4 Fractional baud rate generation

The baud rates for the receiver and transmitter are both set to the same value as programmed in the Mantissa (LINIBRR) and Fraction (LINFBR) registers.

$$\text{Tx/ Rx baud} = \frac{f_{\text{periph_set_1_clk}}}{(16 \times \text{LFDIV})}$$

LFDIV is an unsigned fixed point number. The 12-bit mantissa is coded in the LINIBRR and the fraction is coded in the LINFBR.

The following examples show how to derive LFDIV from LINIBRR and LINFBR register values:

Example 1. Deriving LFDIV from LINIBRR and LINFBR register values

If LINIBRR = 27d and LINFBR = 12d, then

Mantissa (LFDIV) = 27d

Fraction (LFDIV) = 12/16 = 0.75d

Therefore LFDIV = 27.75d

Example 2. Programming LFDIV from LINIBRR and LINFBR register values

To program LFDIV = 25.62d,

LINFBR = 16 × 0.62 = 9.92, nearest real number 10d = 0xA

LINIBRR = mantissa (25.620d) = 25d = 0x19

NOTE

The baud counters are updated with the new value of the baud registers after a write to LINIBRR. Hence the baud register value must not be changed during a transaction. The LINFBR (containing the Fraction bits) must be programmed before the LINIBRR.

NOTE

LFDIV must be greater than or equal to 1.5d, i.e. LINIBRR = 1 and LINFBR = 8. Therefore, the maximum possible baudrate is $f_{\text{periph_set_1_clk}} / 24$.

$$\text{Tx/ Rx baud} = \frac{f_{\text{periph_set_1_clk}}}{(16 \times \text{LFDIV})}$$

LFDIV is an unsigned fixed point number. The 12-bit mantissa is coded in the LINIBRR and the fraction is coded in the LINFBR.

The following examples show how to derive LFDIV from LINIBRR and LINFBR register values:

Example 3. Deriving LFDIV from LINIBRR and LINFBR register values

If LINIBRR = 27d and LINFBR = 12d, then

Mantissa (LFDIV) = 27d

Fraction (LFDIV) = 12/16 = 0.75d

Therefore LFDIV = 27.75d

Example 4. Programming LFDIV from LINIBRR and LINFBR register values

To program LFDIV = 25.62d,

LINFBR = 16 × 0.62 = 9.92, nearest real number 10d = 0xA

LINIBRR = mantissa (25.620d) = 25d = 0x19

NOTE

The baud counters are updated with the new value of the baud registers after a write to LINIBRR. Hence the baud register value must not be changed during a transaction. The LINFBR (containing the Fraction bits) must be programmed before the LINIBRR.

NOTE

LFDIV must be greater than or equal to 1.5d, i.e. LINIBRR = 1 and LINFBR = 8. Therefore, the maximum possible baudrate is $f_{\text{periph_set_1_clk}} / 24$.

Table 312. Error calculation for programmed baud rates

Baud rate	$f_{\text{periph_set_1_clk}} = 64 \text{ MHz}$				$f_{\text{periph_set_1_clk}} = 16 \text{ MHz}$			
	Actual	Value programmed in the baud rate register		% Error = (Calculated – Desired) baud rate / Desired baud rate	Actual	Value programmed in the baud rate register		% Error = (Calculated – Desired) baud rate / Desired baud rate
		LINIBRR	LINFBR			LINIBRR	LINFBR	
2400	2399.97	1666	11	-0.001	2399.88	416	11	-0.005
9600	9599.52	416	11	-0.005	9598.08	104	3	-0.02
10417	10416.7	384	0	-0.003	10416.7	96	0	-0.003
19200	19201.9	208	5	0.01	19207.7	52	1	0.04
57600	57605.8	69	7	0.01	57554	17	6	-0.08
115200	115108	34	12	-0.08	115108	8	11	-0.08
230400	230216	17	6	-0.08	231884	4	5	0.644
460800	460432	8	11	-0.08	457143	2	3	-0.794
921600	927536	4	5	0.644	941176	1	1	2.124

25.5 Operating modes

LINFlex has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, LINFlex is in Sleep mode to reduce power consumption. The software instructs LINFlex to enter Initialization mode or Sleep mode by setting the INIT bit or SLEEP bit in the LINCRI.

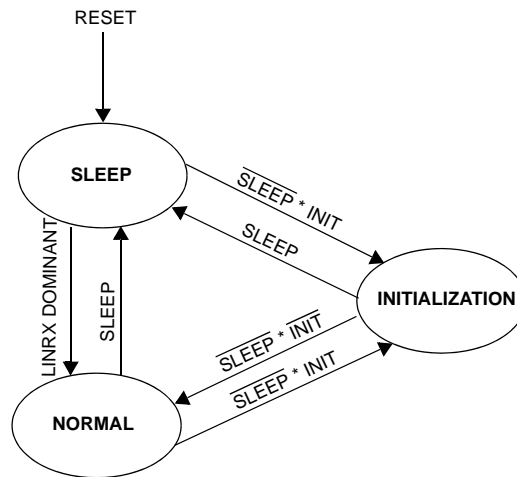


Figure 284. LINFlex operating modes

25.5.1 Initialization mode

The software can be initialized while the hardware is in Initialization mode. To enter this mode the software sets the INIT bit in the LINCRI.

To exit Initialization mode, the software clears the INIT bit.

While in Initialization mode, all message transfers to and from the LIN bus are stopped and the status of the LIN bus output LINTX is recessive (high).

Entering Initialization mode does not change any of the configuration registers.

To initialize the LINFlex controller, the software selects the mode (LIN Master, LIN Slave or UART), sets up the baud rate register and, if LIN Slave mode with filter activation is selected, initializes the identifier list.

25.5.2 Normal mode

Once initialization is complete, software clears the INIT bit in the LINCRI to put the hardware into Normal mode.

25.5.3 Low power mode (Sleep)

To reduce power consumption, LINFlex has a low power mode called Sleep mode. To enter Sleep mode, software sets the SLEEP bit in the LINC1. In this mode, the LINFlex clock is stopped. Consequently, the LINFlex will not update the status bits but software can still access the LINFlex registers.

LINFlex can be awakened (exit Sleep mode) either by software clearing the SLEEP bit or on detection of LIN bus activity if automatic wake-up mode is enabled (AWUM bit is set).

On LIN bus activity detection, hardware automatically performs the wake-up sequence by clearing the SLEEP bit if the AWUM bit in the LINC1 is set. To exit from Sleep mode if the AWUM bit is cleared, software clears the SLEEP bit when a wake-up event occurs.

25.6 Test modes

Two test modes are available to the user: Loop Back mode and Self Test mode. They can be selected by the LBKM and SFTM bits in the LINC1. These bits must be configured while LINFlex is in Initialization mode. Once one of the two test modes has been selected, LINFlex must be started in Normal mode.

25.6.1 Loop Back mode

LINFlex can be put in Loop Back mode by setting the LBKM bit in the LINC1. In Loop Back mode, the LINFlex treats its own transmitted messages as received messages.

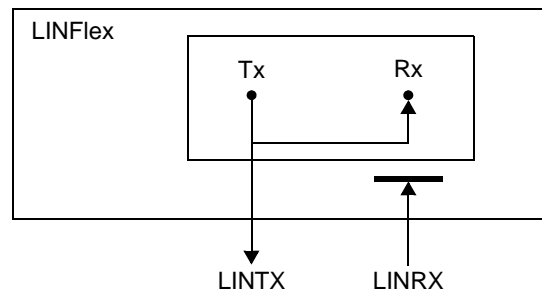


Figure 285. LINFlex in loop back mode

This mode is provided for self test functions. To be independent of external events, the LIN core ignores the LINRX signal. In this mode, the LINFlex performs an internal feedback from its Tx output to its Rx input. The actual value of the LINRX input pin is disregarded by the LINFlex. The transmitted messages can be monitored on the LINTX pin.

25.6.2 Self Test mode

LINFlex can be put in Self Test mode by setting the LBKM and SFTM bits in the LINC1. This mode can be used for a “Hot Self Test”, meaning the LINFlex can be tested as in Loop Back mode but without affecting a running LIN system connected to the LINTX and LINRX pins. In this mode, the LINRX pin is disconnected from the LINFlex and the LINTX pin is held recessive.

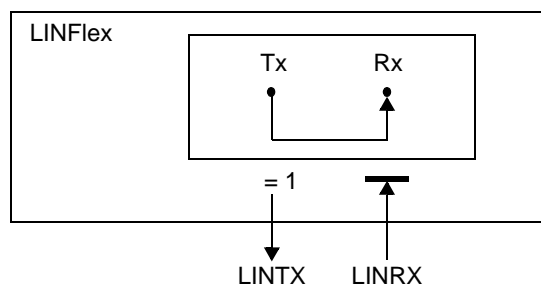


Figure 286. LINFlex in self test mode

25.7 Memory map and registers description

25.7.1 Memory map

The base addresses for the LINFlex modules are as follows:

- 0xFFE4_0000 (LINFlex_0)
- 0xFFE4_4000 (LINFlex_1)

Table 313 shows the LINFlex memory map.

Table 313. LINFlex_0 memory map

Offset from LINFLEX_BASE	Register	Reset value	Location
0x0000	LIN control register 1 (LINCRR1)	0x0000_0082	on page 575
0x0004	LIN interrupt enable register (LINIER)	0x0000_0000	on page 578
0x0008	LIN status register (LINSR)	0x0000_0040	on page 579
0x000C	LIN error status register (LINESR)	0x0000_0000	on page 582
0x0010	UART mode control register (UARTCR)	0x0000_0000	on page 583
0x0014	UART mode status register (UARTSR)	0x0000_0000	on page 585
0x0018	LIN timeout control status register (LINTCSR)	0x0000_0040	on page 587
0x001C	LIN output compare register (LINOOCR)	0x0000_FFFF	on page 588
0x0020	LIN timeout control register (LINTOCR)	0x0000_0E4C	on page 588
0x0024	LIN fractional baud rate register (LINFBR)	0x0000_0000	on page 589
0x0028	LIN integer baud rate register (LINIBRR)	0x0000_0000	on page 589
0x002C	LIN checksum field register (LINCFR)	0x0000_0000	on page 589
0x0030	LIN control register 2 (LINCRR2)	0x0000_0000	on page 589
0x0034	Buffer identifier register (BIDR)	0x0000_0000	on page 592
0x0038	Buffer data register least significant (BDRL)	0x0000_0000	on page 593

Table 313. LINFlex_0 memory map (continued)

Offset from LINFLEX_BASE	Register	Reset value	Location
0x003C	Buffer data register most significant (BDRM)	0x0000_0000	on page 594
0x0040	Identifier filter enable register (IFER)	0x0000_0000	on page 594
0x0044	Identifier filter match index (IFMI)	0x0000_0000	on page 596
0x0048	Identifier filter mode register (IFMR)	0x0000_0000	on page 596
0x004C	Identifier filter control register 0 (IFCR0)	0x0000_0000	on page 598
0x0050	Identifier filter control register 1 (IFCR1)	0x0000_0000	on page 599
0x0054	Identifier filter control register 2 (IFCR2)	0x0000_0000	on page 599
0x0058	Identifier filter control register 3 (IFCR3)	0x0000_0000	on page 599
0x005C	Identifier filter control register 4 (IFCR4)	0x0000_0000	on page 599
0x0060	Identifier filter control register 5 (IFCR5)	0x0000_0000	on page 599
0x0064	Identifier filter control register 6 (IFCR6)	0x0000_0000	on page 599
0x0068	Identifier filter control register 7 (IFCR7)	0x0000_0000	on page 599
0x006C	Identifier filter control register 8 (IFCR8)	0x0000_0000	on page 599
0x0070	Identifier filter control register 9 (IFCR9)	0x0000_0000	on page 599
0x0074	Identifier filter control register 10 (IFCR10)	0x0000_0000	on page 599
0x0078	Identifier filter control register 11 (IFCR11)	0x0000_0000	on page 599
0x007C	Identifier filter control register 12 (IFCR12)	0x0000_0000	on page 599
0x0080	Identifier filter control register 13 (IFCR13)	0x0000_0000	on page 599
0x0084	Identifier filter control register 14 (IFCR14)	0x0000_0000	on page 599
0x0088	Identifier filter control register 15 (IFCR15)	0x0000_0000	on page 599
0x008C–0x000F	Reserved		

Table 314. LINFlex_1 memory map

Offset from LINFLEX_BASE	Register	Reset value	Location
0x0000	LIN control register 1 (LINCR1)	0x0000_0092	on page 575
0x0004	LIN interrupt enable register (LINIER)	0x0000_0000	on page 578
0x0008	LIN status register (LINSR)	0x0000_0040	on page 579
0x000C	LIN error status register (LINESR)	0x0000_0000	on page 582
0x0010	UART mode control register (UARTCR)	0x0000_0000	on page 583
0x0014	UART mode status register (UARTSR)	0x0000_0000	on page 585
0x0018	LIN timeout control status register (LINTCSR)	0x0000_0040	on page 587

Table 314. LINFlex_1 memory map

Offset from LINFLEX_BASE	Register	Reset value	Location
0x001C	LIN output compare register (LINOCR)	0x0000_FFFF	on page 588
0x0020	LIN timeout control register (LINTOCR)	0000_0E1Ch	on page 588
0x0024	LIN fractional baud rate register (LINFBR)	0x0000_0000	on page 589
0x0028	LIN integer baud rate register (LINIBRR)	0x0000_0000	on page 589
0x002C	LIN checksum field register (LINCFR)	0x0000_0000	on page 589
0x0030	LIN control register 2 (LINCRR2)	0x0000_0000	on page 589
0x0034	Buffer identifier register (BIDR)	0x0000_0000	on page 592
0x0038	Buffer data register least significant (BDRL)	0x0000_0000	on page 593
0x003C	Buffer data register most significant (BDRM)	0x0000_0000	on page 594
0x0040	Identifier filter enable register (IFER)	0x0000_0000	on page 594
0x0044	Identifier filter match index (IFMI)	0x0000_0000	on page 596
0x0048	Identifier filter mode register (IFMR)	0x0000_0000	on page 596
0x004C–0x000F	Reserved		

25.7.2 Register description

This section describes in address order all the LINFlex registers. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

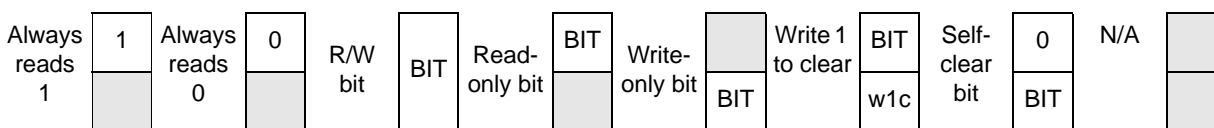


Figure 287. Key to register fields

25.7.2.1 LIN control register 1 (LINC1)

Address: Base + 0x0000

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD	CFD	LASE	AWUM	MBL[0:3]				BF	SFTM	LBKM	MME	SBDT	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0 ¹	0	0	1	0

Figure 288. LIN control register 1 (LINC1)
¹ For Linflex_1, it is 1.

Table 315. LINC1 field descriptions

Field	Description
0:15	Reserved
CCD 16	Checksum calculation disable This bit disables the checksum calculation (see Table 316). 0 Checksum calculation is done by hardware. When this bit is 0, the LINC1R is read-only. 1 Checksum calculation is disabled. When this bit is set the LINC1R is read/write. User can program this register to send a software-calculated CRC (provided CFD is 0). Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
CFD 17	Checksum field disable This bit disables the checksum field transmission (see Table 316). 0 Checksum field is sent after the required number of data bytes is sent. 1 No checksum field is sent. Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
LASE 18	LIN Slave Automatic Resynchronization Enable 0 Automatic resynchronization disable. 1 Automatic resynchronization enable. Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
AWUM 19	Automatic Wake-Up Mode This bit controls the behavior of the LINFlex hardware during Sleep mode. 0 The Sleep mode is exited on software request by clearing the SLEEP bit of the LINC1R. 1 The Sleep mode is exited automatically by hardware on LINRX dominant state detection. The SLEEP bit of the LINC1R is cleared by hardware whenever WUF bit in the LINSR is set. Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
MBL[0:3] 20:23	LIN Master Break Length These bits indicate the Break length in Master mode (see Table 317). Note: These bits can be written in Initialization mode only. They are read-only in Normal or Sleep mode.

Table 315. LINC1 field descriptions (continued)

Field	Description
BF 24	<p>Bypass filter</p> <p>0 No interrupt if identifier does not match any filter.</p> <p>1 An RX interrupt is generated on identifier not matching any filter.</p> <p>Note:</p> <ul style="list-style-type: none"> If no filter is activated, this bit is reserved. This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.
SFTM 25	<p>Self Test Mode</p> <p>This bit controls the Self Test mode. For more details, see Section 25.6.2, “Self Test mode”.</p> <p>0 Self Test mode disable.</p> <p>1 Self Test mode enable.</p> <p>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.</p>
LBKM 26	<p>Loop Back Mode</p> <p>This bit controls the Loop Back mode. For more details see Section 25.6.1, “Loop Back mode”.</p> <p>0 Loop Back mode disable.</p> <p>1 Loop Back mode enable.</p> <p>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode</p>
MME 27	<p>Master Mode Enable</p> <p>0 Slave mode enable.</p> <p>1 Master mode enable.</p> <p>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.</p>
SBDT 28	<p>Slave Mode Break Detection Threshold</p> <p>0 11-bit break.</p> <p>1 10-bit break.</p> <p>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.</p>
RBLM 29	<p>Receive Buffer Locked Mode</p> <p>0 Receive Buffer not locked on overrun. Once the Slave Receive Buffer is full the next incoming message overwrites the previous one.</p> <p>1 Receive Buffer locked against overrun. Once the Receive Buffer is full the next incoming message is discarded.</p> <p>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode.</p>
SLEEP 30	<p>Sleep Mode Request</p> <p>This bit is set by software to request LINFlex to enter Sleep mode.</p> <p>This bit is cleared by software to exit Sleep mode or by hardware if the AWUM bit in LINC1 and the WUF bit in LINSR are set (see Table 318).</p>
INIT 31	<p>Initialization Request</p> <p>The software sets this bit to switch hardware into Initialization mode. If the SLEEP bit is reset, LINFlex enters Normal mode when clearing the INIT bit (see Table 318).</p>

Table 316. Checksum bits configuration

CFD	CCD	LINCFR	Checksum sent
1	1	Read/Write	None
1	0	Read-only	None
0	1	Read/Write	Programmed in LINCFR by bits CF[0:7]
0	0	Read-only	Hardware calculated

Table 317. LIN master break length selection

MBL[0:3]	Length
0000	10-bit
0001	11-bit
0010	12-bit
0011	13-bit
0100	14-bit
0101	15-bit
0110	16-bit
0111	17-bit
1000	18-bit
1001	19-bit
1010	20-bit
1011	21-bit
1100	22-bit
1101	23-bit
1110	36-bit
1111	50-bit

Table 318. Operating mode selection

SLEEP	INIT	Operating mode
1	0	Sleep (reset value)
x	1	Initialization
0	0	Normal

25.7.2.2 LIN interrupt enable register (LINIER)

Address: Base + 0x0004

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0	0	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 289. LIN interrupt enable register (LINIER)

Table 319. LINIER field descriptions

Field	Description
0:15	Reserved
SZIE 16	Stuck at Zero Interrupt Enable 0 No interrupt when SZF bit in LINESR or UARTSR is set. 1 Interrupt generated when SZF bit in LINESR or UARTSR is set.
OCIE 17	Output Compare Interrupt Enable 0 No interrupt when OCF bit in LINESR or UARTSR is set. 1 Interrupt generated when OCF bit in LINESR or UARTSR is set.
BEIE 18	Bit Error Interrupt Enable 0 No interrupt when BEF bit in LINESR is set. 1 Interrupt generated when BEF bit in LINESR is set.
CEIE 19	Checksum Error Interrupt Enable 0 No interrupt on Checksum error. 1 Interrupt generated when checksum error flag (CEF) in LINESR is set.
HEIE 20	Header Error Interrupt Enable 0 No interrupt on Break Delimiter error, Synch Field error, Identifier field error. 1 Interrupt generated on Break Delimiter error, Synch Field error, Identifier field error.
21:22	Reserved
FEIE 23	Framing Error Interrupt Enable 0 No interrupt on Framing error. 1 Interrupt generated on Framing error.
BOIE 24	Buffer Overrun Interrupt Enable 0 No interrupt on Buffer overrun. 1 Interrupt generated on Buffer overrun.

Table 319. LINIER field descriptions (continued)

Field	Description
LSIE 25	LIN State Interrupt Enable 0 No interrupt on LIN state change. 1 Interrupt generated on LIN state change. This interrupt can be used for debugging purposes. It has no status flag but is reset when writing '1111' into LINS[0:3] in the LINSR.
WUIE 26	Wake-up Interrupt Enable 0 No interrupt when WUF bit in LINSR or UARTSR is set. 1 Interrupt generated when WUF bit in LINSR or UARTSR is set.
DBFIE 27	Data Buffer Full Interrupt Enable 0 No interrupt when buffer data register is full. 1 Interrupt generated when data buffer register is full.
DBEIE 28	Data Buffer Empty Interrupt Enable 0 No interrupt when buffer data register is empty. 1 Interrupt generated when data buffer register is empty.
DRIE 29	Data Reception Complete Interrupt Enable 0 No interrupt when data reception is completed. 1 Interrupt generated when data received flag (DRF) in LINSR or UARTSR is set.
DTIE 30	Data Transmitted Interrupt Enable 0 No interrupt when data transmission is completed. 1 Interrupt generated when data transmitted flag (DTF) is set in LINSR or UARTSR.
HRIE 31	Header Received Interrupt Enable 0 No interrupt when a valid LIN header has been received. 1 Interrupt generated when a valid LIN header has been received, that is, HRF bit in LINSR is set.

25.7.2.3 LIN status register (LINSR)

Address: Base + 0x0008

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0	0	RMB	0	RPS	WUF	DBFF	DBFL	DRF	DTF	HRF	
W	w1c	w1c	w1c	w1c			w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 290. LIN status register (LINSR)

Table 320. LINSR field descriptions

Field	Description
LINS	<p>LIN state LIN mode states description 0000: Sleep mode LINFlex is in Sleep mode to save power consumption. 0001: Initialization mode LINFlex is in Initialization mode. 0010: Idle This state is entered on several events:</p> <ul style="list-style-type: none"> • SLEEP bit and INIT in LINCR1 register have been cleared by software, • A falling edge has been received on RX pin and AWUM bit is set, • The previous frame reception or transmission has been completed or aborted. <p>0011: Break In Slave mode, a falling edge followed by a dominant state has been detected. Receiving Break. Note: In Slave mode, in case of error new LIN state can be either Idle or Break depending on last bit state. If last bit is dominant new LIN state is Break, otherwise Idle. In Master mode, Break transmission ongoing.</p> <p>0100: Break Delimiter In Slave mode, a valid Break has been detected. Refer to LINCR1 register for break length configuration (10-bit or 11-bit). Waiting for a rising edge. In Master mode, Break transmission has been completed. Break Delimiter transmission is ongoing.</p> <p>0101: Synch Field In Slave mode, a valid Break Delimiter has been detected (recessive state for at least one bit time). Receiving Synch Field. In Master mode, Synch Field transmission is ongoing.</p> <p>0110: Identifier Field In Slave mode, a valid Synch Field has been received. Receiving ID Field. In Master mode, identifier transmission is ongoing.</p> <p>0111: Header reception/transmission completed In Slave mode, a valid header has been received and identifier field is available in the BIDR register. In Master mode, header transmission is completed.</p> <p>1000: Data reception/transmission Response reception/transmission is ongoing.</p> <p>1001: Checksum Data reception/transmission completed. Checksum reception/transmission ongoing. Note: LIN state bits (LINS3:0) should be used in the LIN mode only.</p>
RMB	<p>Release Message Buffer 0: Buffer is free 1: Buffer ready to be read by software. This bit must be cleared by software after reading data received in the buffer. This bit is cleared by hardware in Initialization mode.</p>
RBSY	<p>Receiver Busy Flag 0: Receiver is Idle 1: Reception ongoing Note: In Slave mode, after header reception, if DIR bit in BIDR is reset and reception starts then this bit is set. In this case, user cannot set DTRQ bit in LINCR2.</p>
RPS	<p>LIN receive pin state This bit reflects the current status of LINRX pin for diagnostic purposes.</p>

Table 320. LINSR field descriptions (continued)

Field	Description
WUF	Wake-up Flag This bit is set by hardware and indicates to the software that LINFlex has detected a falling edge on the LINRX pin when <ul style="list-style-type: none"> • slave is in Sleep mode, • master is in Sleep mode or idle state. This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt is generated if WUIE bit in LINIER is set.
DBFF	Data Buffer Full Flag This bit is set by hardware and indicates the buffer is full. It is set only when receiving extended frames (DFL > 7). This bit must be cleared by software. It is reset by hardware in Initialization mode.
DBEF	Data Buffer Empty Flag This bit is set by hardware and indicates the buffer is empty. It is set only when transmitting extended frames (DFL > 7). This bit must be cleared by software, once buffer has been filled again, in order to start transmission. This bit is reset by hardware in Initialization mode.
DRF	Data Reception Completed Flag This bit is set by hardware and indicates the data reception is completed. This bit must be cleared by software. It is reset by hardware in Initialization mode. Note: This flag is not set in case of bit error or framing error.
DTF	Data Transmission Completed Flag This bit is set by hardware and indicates the data transmission is completed. This bit must be cleared by software. It is reset by hardware in Initialization mode. Note: This flag is not set in case of bit error if IOBE bit is reset.
HRF	Header Reception Flag This bit is set by hardware and indicates a valid header reception is completed. This bit must be cleared by software. This bit is reset by hardware in Initialization mode and at end of completed or aborted frame. Note: If filters are enabled, this bit is set only when identifier software filtering is required, that is to say: <ul style="list-style-type: none"> • all filters are inactive and BF bit in LINCR1 is set • no match in any filter and BF bit in LINCR1 is set • TX filter match

25.7.2.4 LIN error status register (LINESR)

Address: Base + 0x000C

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SFEF	BDEF	IDPEF	FEF	BOF	0	0	0	0	0	0	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 291. LIN error status register (LINESR)

Table 321. LINESR field descriptions

Field	Description
SZF	Stuck at zero Flag This bit is set by hardware when the bus is dominant for more than a 100-bit time. It is cleared by software.
OCF	Output Compare Flag 0: No output compare event occurred 1: The content of the counter has matched the content of OC1[0:7] or OC2[0:7] in LINOCCR. If this bit is set and IOT bit in LINTCSR is set, LINFlex moves to Idle state. If LTOM bit in LINTCSR register is set then OCF is reset by hardware in Initialization mode. If LTOM bit is reset, then OCF maintains its status whatever the mode is.
BEF	Bit Error Flag This bit is set by hardware and indicates to the software that LINFlex has detected a bit error. This error can occur during response field transmission (Slave and Master modes) or during header transmission (in Master mode). This bit is cleared by software.
CEF	Checksum error Flag This bit is set by hardware and indicates that the received checksum does not match the hardware calculated checksum. This bit is cleared by software. Note: This bit is never set if CCD or CFD bit in LINCR1 register is set.
SFEF	Synch Field Error Flag This bit is set by hardware and indicates that a Synch Field error occurred (inconsistent Synch Field).
BDEF	Break Delimiter Error Flag This bit is set by hardware and indicates that the received Break Delimiter is too short (less than one bit time).
IDPEF	Identifier Parity Error Flag This bit is set by hardware and indicates that a Identifier Parity error occurred. Note: Header interrupt is triggered when SFEF or BDEF or IDPEF bit is set and HEIE bit in LINIER is set.

Table 321. LINESR field descriptions (continued)

Field	Description
FEF	Framing Error Flag This bit is set by hardware and indicates to the software that LINFlex has detected a framing error (invalid stop bit). This error can occur during reception of any data in the response field (Master or Slave mode) or during reception of Synch Field or Identifier Field in Slave mode.
BOF	Buffer Overrun Flag This bit is set by hardware when a new data byte is received and the buffer full flag is not cleared. If RBLM in LINCR1 is set then the new byte received is discarded. If RBLM is reset then the new byte overwrites the buffer. It can be cleared by software.
-	Reserved (read returns a zero)
NF	Noise Flag This bit is set by hardware when noise is detected on a received character. This bit is cleared by software.

25.7.2.5 UART mode control register (UARTCR)

Address: Base + 0x0010

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0			0	0	0	0						
W		TDFL[0:1]			RDFL[0:1]						RXEN	TXEN	OP	PCE	WL	UART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 292. UART mode control register (UARTCR)
Table 322. UARTCR field descriptions

Field	Description
0:16	Reserved
TDFL[0:1] 17:18	Transmitter Data Field length These bits set the number of bytes to be transmitted in UART mode. These bits can be programmed only when the UART bit is set. TDFL[0:1] = Transmit buffer size – 1. 00 Transmit buffer size = 1. 01 Transmit buffer size = 2. 10 Transmit buffer size = 3. 11 Transmit buffer size = 4.
19	Reserved

Table 322. UARTCR field descriptions (continued)

Field	Description
RDFL[0:1] 20:21	Receiver Data Field length These bits set the number of bytes to be received in UART mode. These bits can be programmed only when the UART bit is set. RDFL[0:1] = Receive buffer size – 1. 00 Receive buffer size = 1. 01 Receive buffer size = 2. 10 Receive buffer size = 3. 11 Receive buffer size = 4.
22:25	Reserved
RXEN 26	Receiver Enable 0 Receiver disable. 1 Receiver enable. This bit can be programmed only when the UART bit is set.
TXEN 27	Transmitter Enable 0 Transmitter disable. 1 Transmitter enable. This bit can be programmed only when the UART bit is set. Note: Transmission starts when this bit is set and when writing DATA0 in the BDRL register.
OP 28	Odd Parity 0 Sent parity is even. 1 Sent parity is odd. This bit can be programmed in Initialization mode only when the UART bit is set.
PCE 29	Parity Control Enable 0 Parity transmit/check disable. 1 Parity transmit/check enable. This bit can be programmed in Initialization mode only when the UART bit is set.
WL 30	Word Length in UART mode 0 7-bit data + parity bit. 1 8-bit data (or 9-bit if PCE is set). This bit can be programmed in Initialization mode only when the UART bit is set.
UART 31	UART mode enable 0 LIN mode. 1 UART mode. This bit can be programmed in Initialization mode only.

25.7.2.6 UART mode status register (UARTSR)

Address: Base + 0x0014

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE3	PE2	PE1	PE0	RMB	FEF	BOF	RPS	WUF	0	TO	DRFRFE	DTFTFF	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 293. UART mode status register (UARTSR)
Table 323. UARTSR field descriptions

Field	Description
SZF	Stuck at zero Flag This bit is set by hardware when the bus is dominant for more than a 100-bit time. It is cleared by software.
OCF	OCF Output Compare Flag 0: No output compare event occurred 1: The content of the counter has matched the content of OC1[0:7] or OC2[0:7] in LINOCR. An interrupt is generated if the OCIE bit in LINIER register is set.
PE3	Parity Error Flag Rx3 This bit indicates if there is a parity error in the corresponding received byte (Rx3). No interrupt is generated if this error occurs. 0: No parity error 1: Parity error
PE2	Parity Error Flag Rx2 This bit indicates if there is a parity error in the corresponding received byte (Rx2). No interrupt is generated if this error occurs. 0: No parity error 1: Parity error
PE1	Parity Error Flag Rx1 This bit indicates if there is a parity error in the corresponding received byte (Rx1). No interrupt is generated if this error occurs. 0: No parity error 1: Parity error
PE0	Parity Error Flag Rx0 This bit indicates if there is a parity error in the corresponding received byte (Rx0). No interrupt is generated if this error occurs. 0: No parity error 1: Parity error

Table 323. UARTSR field descriptions (continued)

Field	Description
RMB	Release Message Buffer 0: Buffer is free 1: Buffer ready to be read by software. This bit must be cleared by software after reading data received in the buffer. This bit is cleared by hardware in Initialization mode.
FEF	Framing Error Flag This bit is set by hardware and indicates to the software that LINFlex has detected a framing error (invalid stop bit).
BOF	FIFO/buffer overrun flag This bit is set by hardware when a new data byte is received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode, this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message is discarded regardless of the value of LINCR1[RBLM]. If LINCR1[RBLM] = 1, the new byte received is discarded. If LINCR1[RBLM] = 0, the new byte overwrites buffer. This field can be cleared by writing a 1 to it. An interrupt is generated if LINIER[BOIE] is set.
RPS	LIN Receive Pin State This bit reflects the current status of LINRX pin for diagnostic purposes.
WUF	Wake-up Flag This bit is set by hardware and indicates to the software that LINFlex has detected a falling edge on the LINRX pin in Sleep mode. This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt i generated if WUIE bit in LINIER is set.
TO	Timeout The LINFlex controller sets this field when a UART timeout occurs — that is, when the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). This field should be cleared by software. The GCR[SR] field should be used to reset the receiver FSM to idle state in case of UART timeout for UART reception depending on the application both in buffer and FIFO mode. An interrupt is generated when LINIER[DBEIETOIE] is set on the Error interrupt line in UART mode.
DRFRFE	Data reception completed flag / Rx FIFO empty flag The LINFlex controller sets this field as follows: <ul style="list-style-type: none"> In UART buffer mode (RFBM = 0), it indicates that the number of bytes programmed in RDFL has been received. This field should be cleared by software. An interrupt is generated if LINIER[DRIE] is set. This field is set in case of framing error, parity error, or overrun. This field reflects the same value as in LINESR when in Initialization mode and UART bit is set. In UART FIFO mode (RFBM = 1), it indicates that the Rx FIFO is empty. This field is a read-only field used internally by the DMA Rx interface.
DTFTFF	Data transmission completed flag / Tx FIFO full flag The LINFlex controller sets this field as follows: <ul style="list-style-type: none"> In UART buffer mode (TFBM = 0), it indicates that the data transmission is completed. This field should be cleared by software. An interrupt is generated if LINIER[DTIE] is set. This field reflects the same value as in LINESR when in Initialization mode and UART bit is set. In UART FIFO mode (TFBM = 1), it indicates that the Tx FIFO is full. This field is a read-only field used internally by the DMA Tx interface.
NF	Noise Flag This bit is set by hardware when noise is detected on a received character. This bit is cleared by software.

25.7.2.7 LIN timeout control status register (LINTCSR)

Address: Base + 0x0018

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0				CNT[0:7]							
W						LTOM	IOT	TOCE								
Reset	0	0	0	0	0	0		0	0	0		0	0	0	0	0

Figure 294. LIN timeout control status register (LINTCSR)

Table 324. LINTCSR field descriptions

Field	Description
0:20	Reserved
LTOM 21	LIN timeout mode 0 LIN timeout mode (header, response and frame timeout detection). 1 Output compare mode. This bit can be set/cleared in Initialization mode only.
IOT 22	Idle on Timeout 0 LIN state machine not reset to Idle on timeout event. 1 LIN state machine reset to Idle on timeout event. This bit can be set/cleared in Initialization mode only.
TOCE 23	Timeout counter enable 0 Timeout counter disable. OCF bit in LINESR or UARTSR is not set on an output compare event. 1 Timeout counter enable. OCF bit is set if an output compare event occurs. TOCE bit is configurable by software in Initialization mode. If LIN state is not Init and if timer is in LIN timeout mode, then hardware takes control of TOCE bit.
CNT[0:7] 24:31	Counter Value These bits indicate the LIN Timeout counter value.

25.7.2.8 LIN output compare register (LINOCR)

Address: Base + 0x001C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OC2[0:7] ¹								OC1[0:7] ¹							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

¹ If LTOM in the LINTCSR register is set, these bits are read-only.

Figure 295. LIN output compare register (LINOCR)

Table 325. LINOCR field descriptions

Field	Description
0:15	Reserved
OC2[0:7] 16:23	Output compare 2 value These bits contain the value to be compared to the value of bits CNT[0:7] in LINTCSR.
OC1[0:7] 24:31	Output compare 1 value These bits contain the value to be compared to the value of bits CNT[0:7] in LINTCSR.

25.7.2.9 LIN timeout control register (LINTOCR)

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

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	RTO[0:3]				0	HTO[0:6]							
W																	
Reset	0	0	0	0	1	1	1	0	0	0	1	0	1	1	0	0	

Figure 296. LIN timeout control register (LINTOCR)

Table 326. LINTOCR field descriptions

Field	Description
0:19	Reserved
RTO[0:3] 20:23	Response timeout value This register contains the response timeout duration (in bit time) for 1 byte. The reset value is 0xE = 14, corresponding to $T_{Response_Maximum} = 1.4 \times T_{Response_Nominal}$
24	Reserved
HTO[0:6] 25:31	Header timeout value This register contains the header timeout duration (in bit time). This value does not include the Break and the Break Delimiter. The reset value is the 0x2C = 44, corresponding to $T_{Header_Maximum}$. Setting MME bit in LINSR changes HTO[0:6] value to 0x1C = 28. HTO[0:6] can be written only in Slave mode.

25.7.2.10 LIN fractional baud rate register (LINFBR)

Address: Base + 0x0024

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	DIV_F[0:3]			
R	0	0	0	0	0	0	0	0	0	0	0	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 297. LIN fractional baud rate register (LINFBR)
Table 327. LINFBR field descriptions

Field	Description
0:27	Reserved
DIV_F[0:3] 28:31	Fraction bits of LFDIV The 4 fraction bits define the value of the fraction of the LINFlex divider (LFDIV). Fraction (LFDIV) = Decimal value of DIV_F [0:3] / 16. This register can be written in Initialization mode only.

25.7.2.11 LIN Integer Baud Rate Register (LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBR. It can be programmed only in the initialisation mode.

Address: Base + 0x0028 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	IBR12	IBR11	IBR10	IBR9	IBR8	IBR7	IBR6	IBR5	IBR4	IBR3	IBR2	IBR1	IBR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 298. LIN Integer Baud Rate Register (LINIBRR)

Table 328. LINIBRR field descriptions

Field	Description
IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate.

25.7.2.12 LIN checksum field register (LINCFR)

Address: Base + 0x002C Access: User read/write

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

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

Figure 299. LIN checksum field register (LINCFR)

Table 329. LINCFR field descriptions

Field	Description
CF	Checksum bits When LINCRC1[CCD] is cleared, these bits are read-only. When LINCRC1[CCD] is set, these bits are read/write.

25.7.2.13 LIN control register 2 (LINCRC2)

Address: Base + 0x0030

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	IOBE ¹	IOPE ¹	WUR Q	DDR Q	DTR Q	ABR Q	HTR Q	0	0	0	0	0	0	0	0
W				w1c	w1c	w1c	w1c	w1c								
Reset	0	1	0/1 ²	0	0	0	0	0	0	0	0	0	0	0	0	0

¹ These fields are writable only in Initialization mode (LINCRC1[INIT] = 1).

² Resets to 1 in Slave mode and to 0 in Master mode

Figure 300. LIN control register 2 (LINCRC2)

Table 330. LINCRC2 field descriptions

Field	Description
IOBE	Idle on Bit Error 0: Bit error does not reset LIN state machine 1: Bit error reset LIN state machine This bit can be set/cleared in Initialization mode only (LINCRC1[INIT]) = 1.
IOPE	Idle on Identifier Parity Error 0: Identifier Parity error does not reset LIN state machine. 1: Identifier Parity error reset LIN state machine. This bit can be set/cleared in Initialization mode only (LINCRC1[INIT]) = 1.
WURQ	Wake-up Generation Request Setting this bit generates a wake-up pulse. It is reset by hardware when the wake-up character has been transmitted. The character sent is copied from DATA0 in BDRL buffer. Note that this bit cannot be set in Sleep mode. Software has to exit Sleep mode before requesting a wake-up. Bit error is not checked when transmitting the wake-up request.
DDRQ	Data Discard Request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlex has moved to idle state. In Slave mode, this bit can be set only when HRF bit in LINSR is set and identifier did not match any filter.

Table 330. LINCR2 field descriptions (continued)

Field	Description
DTRQ	Data Transmission Request Set by software in Slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when HRF bit in LINSR is set. Cleared by hardware when the request has been completed or aborted or on an error condition. In Master mode, this bit is set by hardware when DIR bit in BIDR is set and header transmission is completed.
ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlex aborts the transmission at the end of the current bit. This bit can also abort a wake-up request. It can also be used in UART mode.
HTRQ	Header Transmission Request Set by software to request the transmission of the LIN header. Cleared by hardware when the request has been completed or aborted. This bit has no effect in UART mode.

25.7.2.14 Buffer identifier register (BIDR)

Address: Base + 0x0034

Access: User read/write

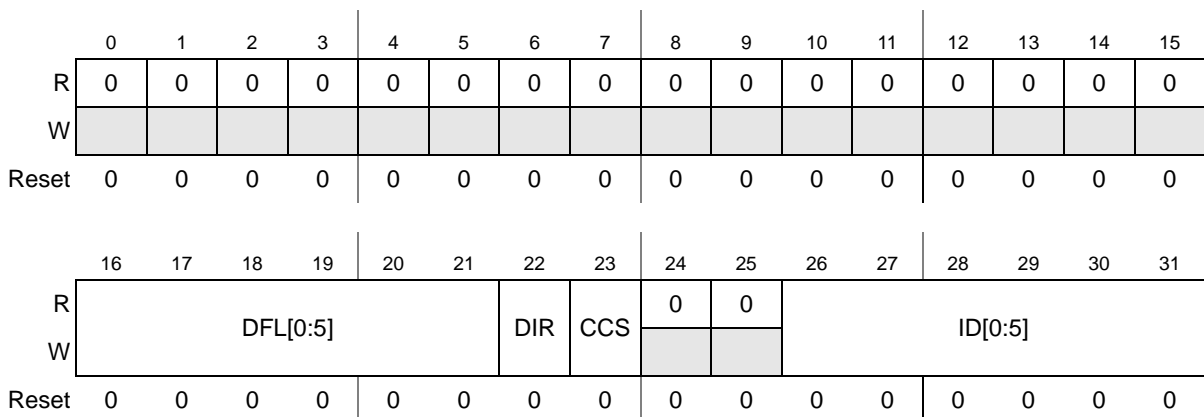


Figure 301. Buffer identifier register (BIDR)

Table 331. BIDR field descriptions

Field	Description
0:15	Reserved
DFL[0:5] 16:21	Data Field Length These bits define the number of data bytes in the response part of the frame. DFL[0:5] = Number of data bytes – 1. Normally, LIN uses only DFL[0:2] to manage frames with a maximum of 8 bytes of data. Identifier filters are compatible with DFL[0:2] only. DFL[3:5] are provided to manage extended frames.

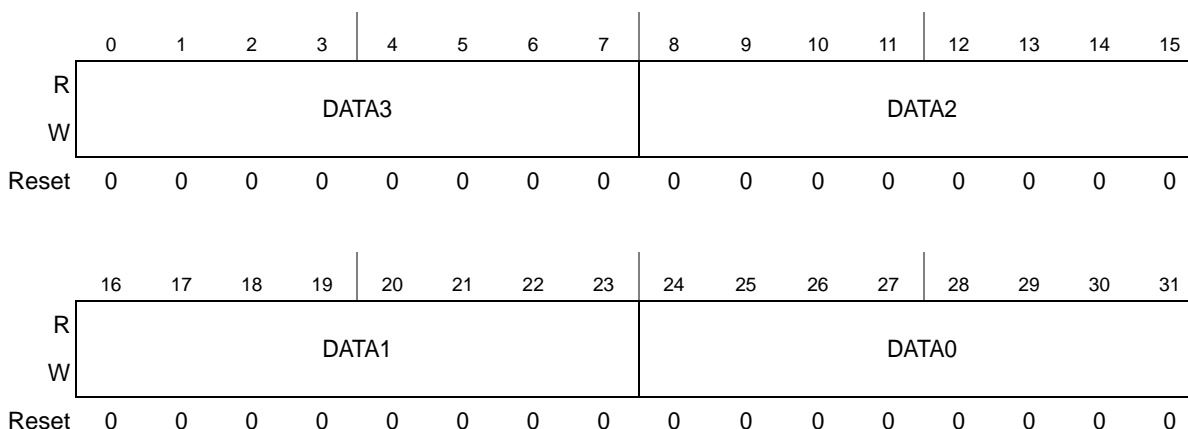
Table 331. BDR field descriptions (continued)

Field	Description
DIR 22	Direction This bit controls the direction of the data field. 0 LINFlex receives the data and copies them in the BDR registers. 1 LINFlex transmits the data from the BDR registers.
CCS 23	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier. In LIN slave mode (MME bit cleared in LINCR1), this bit must be configured before the header reception. If the slave has to manage frames with 2 types of checksum, filters must be configured.
24:25	Reserved
ID[0:5] 26:31	Identifier Identifier part of the identifier field without the identifier parity.

25.7.2.15 Buffer data register least significant (BDRL)

Address: Base + 0x0038

Access: User read/write


Figure 302. Buffer data register least significant (BDRL)
Table 332. BDRL field descriptions

Field	Description
DATA3	Data Byte 3 Data byte 3 of the data field
DATA2	Data Byte 2 Data byte 2 of the data field
DATA1	Data Byte 1 Data byte 1 of the data field
DATA0	Data Byte 0 Data byte 0 of the data field

25.7.2.16 Buffer data register most significant (BDRM)

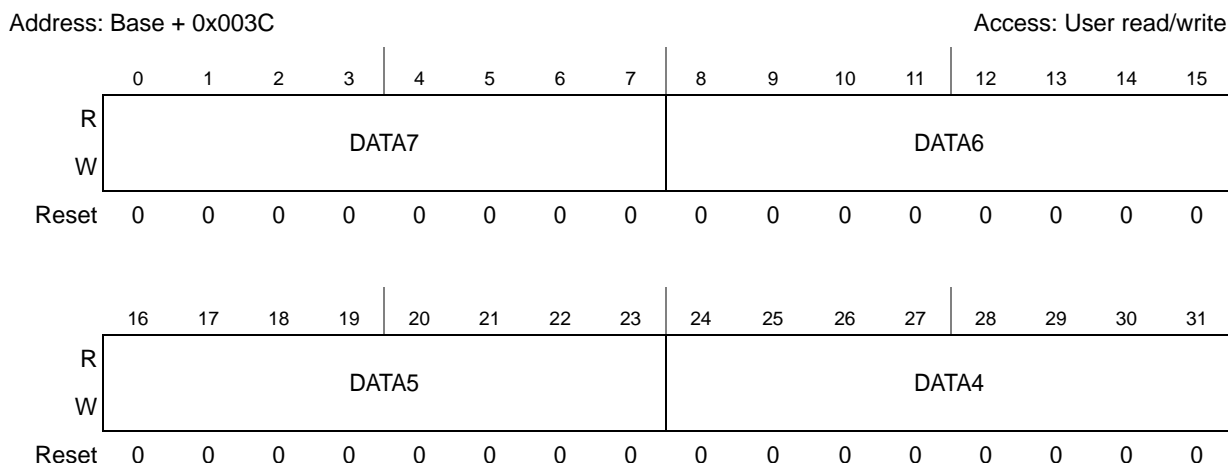


Figure 303. Buffer data register most significant (BDRM)

Table 333. BDRM field descriptions

Field	Description
DATA7	Data Byte 7 Data byte 7 of the data field
DATA6	Data Byte 6 Data byte 6 of the data field
DATA5	Data Byte 5 Data byte 5 of the data field
DATA4	Data Byte 4 Data byte 4 of the data field

25.7.2.17 Identifier filter enable register (IFER)

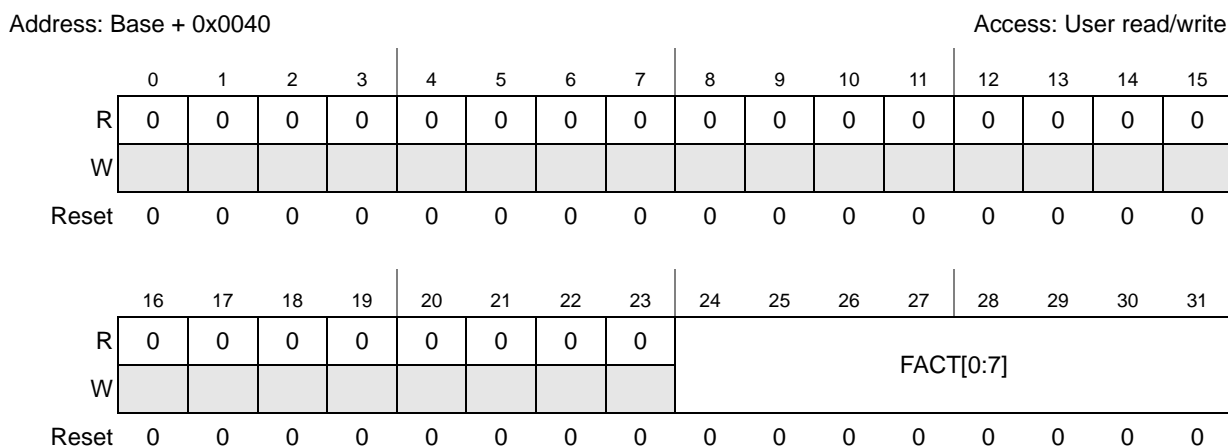


Figure 304. Identifier filter enable register (IFER)

Table 334. IFER field descriptions

Field	Description
0:23	Reserved
FACT[0:7] 24:31	Filter activation 0 Filters $2n$ and $2n + 1$ are activated. 1 Filters $2n$ and $2n + 1$ are deactivated. (Refer to Table 335.) These bits can be set/cleared in Initialization mode only.

Table 335. IFER[FACT] configuration

Bit	Value	Result
FACT[0]	0	Filters 0 and 1 are deactivated.
	1	Filters 0 and 1 are activated.
FACT[1]	0	Filters 2 and 3 are deactivated.
	1	Filters 2 and 3 are activated.
FACT[2]	0	Filters 4 and 5 are deactivated.
	1	Filters 4 and 5 are activated.
FACT[3]	0	Filters 6 and 7 are deactivated.
	1	Filters 6 and 7 are activated.
FACT[4]	0	Filters 8 and 9 are deactivated.
	1	Filters 8 and 9 are activated.
FACT[5]	0	Filters 10 and 11 are deactivated.
	1	Filters 10 and 11 are activated.
FACT[6]	0	Filters 12 and 13 are deactivated.
	1	Filters 12 and 13 are activated.
FACT[7]	0	Filters 14 and 15 are deactivated.
	1	Filters 14 and 15 are activated.

25.7.2.18 Identifier filter match index (IFMI)

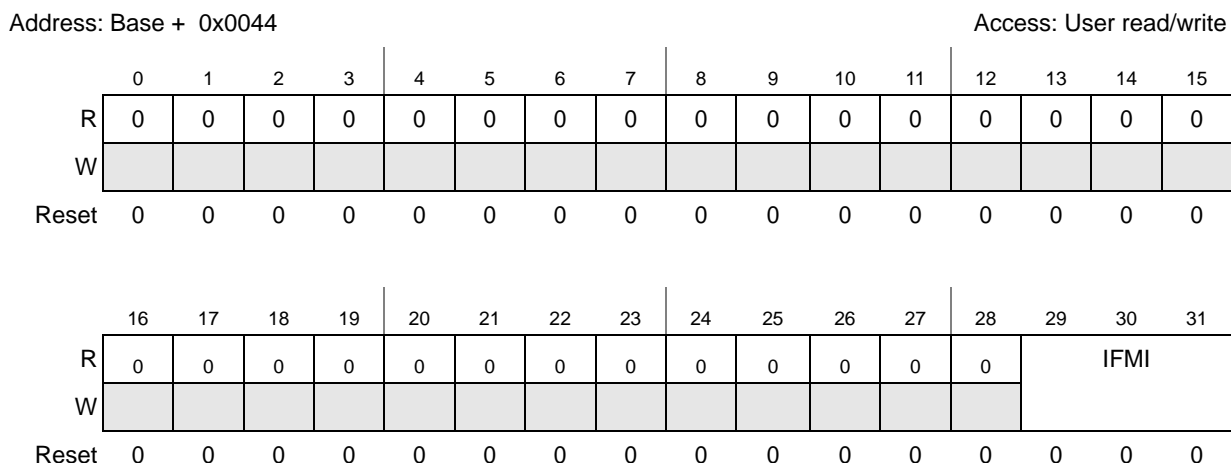


Figure 305. Identifier filter match index (IFMI)

Table 336. IFMI field descriptions

Field	Description
IFMI	Filter match index This register contains the index corresponding to the received ID. It can be used to directly write or read the data in RAM. When no filter matches, IFMI = 0. When Filter n is matching, IFMI = n + 1.

25.7.2.19 Identifier filter mode register (IFMR)

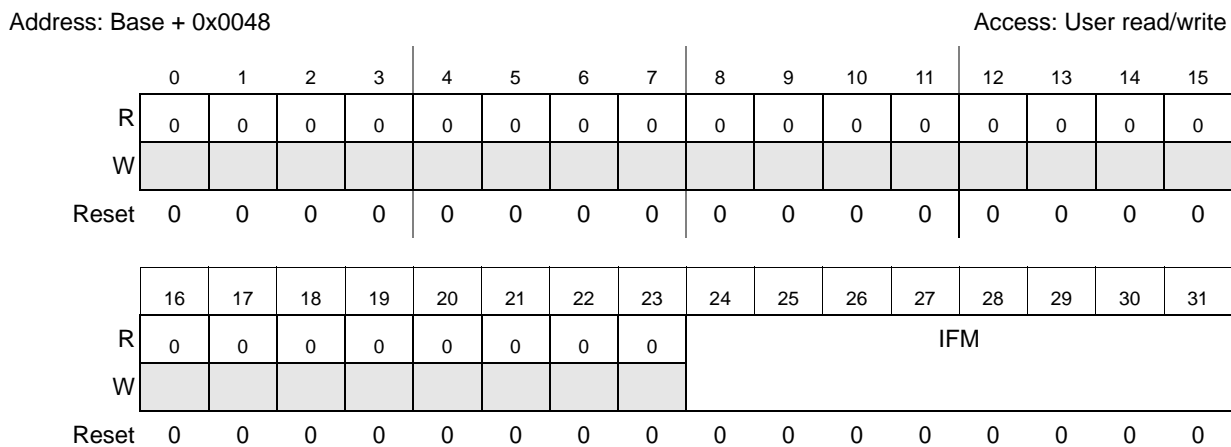


Figure 306. Identifier filter mode register (IFMR)

Table 337. IFMR field descriptions

Field	Description
IFM	Filter mode 0 Filters $2n$ and $2n + 1$ are in identifier list mode. 1 Filters $2n$ and $2n + 1$ are in mask mode (filter $2n + 1$ is the mask for the filter $2n$). (Refer to Table 338 .)

Table 338. IFMR[IFM] configuration

Bit	Value	Result
IFM[0]	0	Filters 0 and 1 are in identifier list mode.
	1	Filters 0 and 1 are in mask mode (filter 1 is the mask for the filter 0).
IFM[1]	0	Filters 2 and 3 are in identifier list mode.
	1	Filters 2 and 3 are in mask mode (filter 3 is the mask for the filter 2).
IFM[2]	0	Filters 4 and 5 are in identifier list mode.
	1	Filters 4 and 5 are in mask mode (filter 5 is the mask for the filter 4).
IFM[3]	0	Filters 6 and 7 are in identifier list mode.
	1	Filters 6 and 7 are in mask mode (filter 7 is the mask for the filter 6).
IFM[4]	0	Filters 8 and 9 are in identifier list mode.
	1	Filters 8 and 9 are in mask mode (filter 9 is the mask for the filter 8).
IFM[5]	0	Filters 10 and 11 are in identifier list mode.
	1	Filters 10 and 11 are in mask mode (filter 11 is the mask for the filter 10).
IFM[6]	0	Filters 12 and 13 are in identifier list mode.
	1	Filters 12 and 13 are in mask mode (filter 13 is the mask for the filter 12).
IFM[7]	0	Filters 14 and 15 are in identifier list mode.
	1	Filters 14 and 15 are in mask mode (filter 15 is the mask for the filter 14).

25.7.2.20 Identifier filter control register (IFCR2n)

n = 0: Address: Base + 0x004C
 n = 1: Address: Base + 0x0054
 n = 2: Address: Base + 0x005C
 n = 3: Address: Base + 0x0064
 n = 4: Address: Base + 0x006C
 n = 5: Address: Base + 0x0074
 n = 6: Address: Base + 0x007C
 n = 7: Address: Base + 0x0084

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0		DFL[0:2]			DIR	CCS	0	0	ID[0:5]				
W					DFL[0:2]			DIR	CCS			w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 307. Identifier filter control register (IFCR2n)

NOTE

This register can be written in Initialization mode only.

Table 339. IFCR2n field descriptions

Field	Description
0:18	Reserved
DFL[0:2] 19:21	Data Field Length These bits define the number of data bytes in the response part of the frame.
DIR 22	Direction This bit controls the direction of the data field. 0 LINFlex receives the data and copies them in the BDRL and BDRM registers. 1 LINFlex transmits the data from the BDRL and BDRM registers.
CCS 23	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier.
24:25	Reserved
ID[0:5] 26:31	Identifier Identifier part of the identifier field without the identifier parity.

25.7.2.21 Identifier filter control register (IFCR2 n + 1)

n = 0: Address: Base + 0x0050

Access: User read/write

n = 1: Address: Base + 0x0058

n = 2: Address: Base + 0x0060

n = 3: Address: Base + 0x0068

n = 4: Address: Base + 0x0070

n = 5: Address: Base + 0x0078

n = 6: Address: Base + 0x0080

n = 7: Address: Base + 0x0088

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0		DFL[0:2]			DIR	CCS	0	0	ID[0:5]				
W					DFL[0:2]			DIR	CCS			w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 308. Identifier filter control register (IFCR2 n + 1)

NOTE

This register can be written in Initialization mode only.

Table 340. IFCR2 n + 1 field descriptions

Field	Description
0:18	Reserved
DFL[0:2] 19:21	Data Field Length These bits define the number of data bytes in the response part of the frame. DFL[0:2] = Number of data bytes – 1.
DIR 22	Direction This bit controls the direction of the data field. 0 LINFlex receives the data and copies them in the BDRL and BDRM registers. 1 LINFlex transmits the data from the BDRL and BDRM registers.
CCS 23	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and earlier.
24:25	Reserved
ID[0:5] 26:31	Identifier Identifier part of the identifier field without the identifier parity

25.8 Functional description

25.8.1 UART mode

The main features in the UART mode are

- Full duplex communication
- 8- or 9-bit data with parity
- 4-byte buffer for reception, 4-byte buffer for transmission
- 8-bit counter for timeout management

8-bit data frames: The 8th bit can be a data or a parity bit. Even/Odd Parity can be selected by the Odd Parity bit in the UARTCR. An even parity is set if the modulo-2 sum of the 7 data bits is 1. An odd parity is cleared in this case.

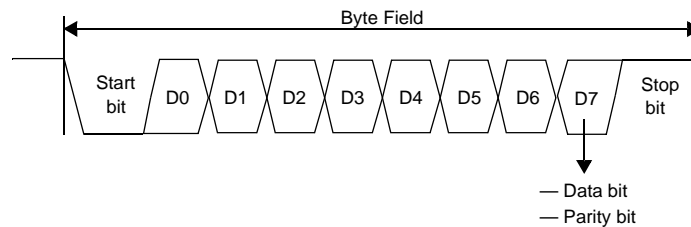


Figure 309. UART mode 8-bit data frame

9-bit frames: The 9th bit is a parity bit. Even/Odd Parity can be selected by the Odd Parity bit in the UARTCR. An even parity is set if the modulo-2 sum of the 7 data bits is 1. An odd parity is cleared in this case.

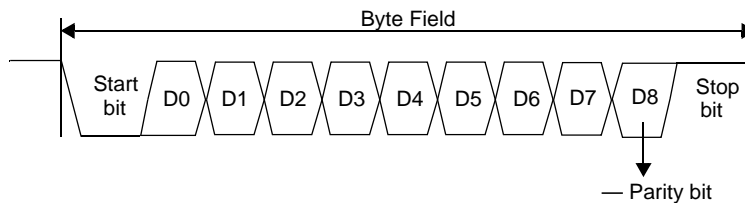


Figure 310. UART mode 9-bit data frame

25.8.1.1 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter as shown in [Table 341](#).

Table 341. Message buffer

Buffer data register	LIN mode		UART mode	
BDRL[0:31]	Transmit/Receive buffer	DATA0[0:7]	Transmit buffer	Tx0
		DATA1[0:7]		Tx1
		DATA2[0:7]		Tx2
		DATA3[0:7]		Tx3
BDRM[0:31]		DATA4[0:7]	Receive buffer	Rx0
		DATA5[0:7]		Rx1
		DATA6[0:7]		Rx2
		DATA7[0:7]		Rx3

25.8.1.2 UART transmitter

In order to start transmission in UART mode, the UART bit and the transmitter enable (TXEN) bit in the UARTCR must be set. Transmission starts when DATA0 (least significant data byte) is programmed. The number of bytes transmitted is equal to the value configured by the TDFL[0:1] bits in the UARTCR (see [Table 322](#)).

The Transmit buffer is 4 bytes, hence a 4-byte maximum transmission can be triggered. Once the programmed number of bytes has been transmitted, the DTF bit is set in UARTSR. If the TXEN bit of UART is reset during a transmission then the current transmission is completed and no further transmission can be invoked.

25.8.1.3 UART receiver

The UART receiver is active as soon as the user exits Initialization mode and sets the RXEN bit in the UARTCR. There is a dedicated 4-byte data buffer for received data bytes. Once the programmed number (RDFL bits) of bytes has been received, the DRF bit is set in UARTSR. If the RXEN bit is reset during a reception then the current reception is completed and no further reception can be invoked until RXEN is set.

If a parity error occurs during reception of any byte, then the corresponding PEx bit in the UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if the FEIE bit in the LINIER is set.

If the last received frame has not been read from the buffer (that is, RMB bit is not reset by the user) then upon reception of the next byte an overrun error occurs (BOF bit in UARTSR is set) and one message will be lost. Which message is lost depends on the configuration of the RBLM bit of LINCR1.

- If the buffer lock function is disabled (RBLM bit in LINCR1 cleared) the last message stored in the buffer is overwritten by the new incoming message. In this case the latest message is always available to the application.
- If the buffer lock function is enabled (RBLM bit in LINCR1 set) the most recent message is discarded and the previous message is available in the buffer.

An interrupt is generated if the BOIE bit in the LINIER is set.

25.8.1.4 Clock gating

The LINFlex clock can be gated from the Mode Entry module (refer to). In UART mode, the LINFlex controller acknowledges a clock gating request once the data transmission and data reception are completed, that is, once the Transmit buffer is empty and the Receive buffer is full.

25.8.2 LIN mode

LIN mode comprises four submodes:

- Master mode
- Slave mode
- Slave mode with identifier filtering
- Slave mode with automatic resynchronization

These submodes are described in the following pages.

25.8.2.1 Master mode

In Master mode the application uses the message buffer to handle the LIN messages. Master mode is selected when the MME bit in LINCR1 is set.

25.8.2.1.1 LIN header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the Master sending a header. The header is transmitted by the Master task while the data is transmitted by the Slave task of a node.

To transmit a header with LINFlex the application must set up the identifier, the data field length and configure the message (direction and checksum type) in the BIDR before requesting the header transmission by setting the HTRQ bit in LINCR2.

25.8.2.1.2 Data transmission (transceiver as publisher)

When the master node is publisher of the data corresponding to the identifier sent in the header, then the slave task of the master has to send the data in the Response part of the LIN frame. Therefore, the application must provide the data to LINFlex before requesting the header transmission. The application stores the data in the message buffer BDR. According to the data field length, LINFlex transmits the data and the checksum. The application uses the CCS bit in the BIDR to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, the DTF bit in the LINSR is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to [Section 25.8.2.1.6, “Error handling”](#)).

It is possible to handle frames with a Response size larger than 8 bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than 8 data bytes, the DBEF bit in the

LINSR is set once the first 8 bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset.

Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is controlled by the DIR bit in the BIDR. When the application sets this bit the response is sent by LINFlex (publisher). Resetting this bit configures the message buffer as subscriber.

25.8.2.1.3 Data reception (transceiver as subscriber)

To receive data from a slave node, the master sends a header with the corresponding identifier. LINFlex stores the data received from the slave in the message buffer and stores the message status in the LINSR.

If the response has been received successfully, the DRF bit in the LINSR is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to [Section 25.8.2.1.6, “Error handling”](#)).

It is possible to handle frames with a Response size larger than 8 bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than 8 data bytes, the DBFF bit in the LINSR is set once the first 8 bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

25.8.2.1.4 Data discard

To discard data from a slave, the DIR bit in the BIDR must be reset and the DDRQ bit in LINCR2 must be set before starting the header transmission.

25.8.2.1.5 Error detection

LINFlex is able to detect and handle LIN communication errors. A code stored in the LIN error status register (LINESR) signals the errors to the software.

In Master mode, the following errors are detected:

- **Bit error:** During transmission, the value read back from the bus differs from the transmitted value.
- **Framing error:** A dominant state has been sampled on the stop bit of the currently received character (synch field, identifier field or data field).
- **Checksum error:** The computed checksum does not match the received one.
- **Response and Frame timeout:** Refer to [Section 25.8.3, “8-bit timeout counter”](#) for more details.

25.8.2.1.6 Error handling

In case of Bit Error detection during transmission, LINFlex stops the transmission of the frame after the corrupted bit. LINFlex returns to idle state and an interrupt is generated if the BEIE bit in the LINIER is set.

During reception, a Framing Error leads LINFlex to discard the current frame. LINFlex returns immediately to idle state. An interrupt is generated if the FEIE bit in the LINIER is set.

During reception, a Checksum Error leads LINFlex to discard the received frame. LINFlex returns to idle state. An interrupt is generated if the CEIE bit in the LINIER is set.

25.8.2.2 Slave mode

In Slave mode the application uses the message buffer to handle the LIN messages. Slave mode is selected when the MME bit in LINCR1 is cleared.

25.8.2.2.1 Data transmission (transceiver as publisher)

When LINFlex receives the identifier, the HRF bit in the LINSR is set and, if the HRIE bit in the LINIER is set, an RX interrupt is generated. The software must read the received identifier in the BIDR, fill the BDR registers, specify the data field length using the DFL[0:2] bits in the BIDR and trigger the data transmission by setting the DTRQ bit in LINCR2.

One or several identifier filters can be configured for transmission by setting the DIR bit in the IFCRx register(s) and activated by setting one or several bits in the IFER.

When at least one identifier filter is configured in transmission and activated, and if the received identifier matches the filter, a specific TX interrupt (instead of an RX interrupt) is generated.

Typically, the application has to copy the data from SRAM locations to the BDAR. To copy the data to the right location, the application has to identify the data by means of the identifier. To avoid this and to ease the access to the SRAM locations, the LINFlex controller provides a Filter Match Index. This index value is the number of the filter that matched the received identifier.

The software can use the index in the IFMI register to directly access the pointer that points to the right data array in the SRAM area and copy this data to the BDAR (see [Figure 312](#)).

Using a filter avoids the software having to configure the direction, the data field length and the checksum type in the BDIR. The software fills the BDAR and triggers the data transmission by setting the DTRQ bit in LINCR2.

If LINFlex cannot provide enough TX identifier filters to handle all identifiers the software has to transmit data for, then a filter can be configured in mask mode (refer to [Section 25.8.2.3, “Slave mode with identifier filtering”](#)) in order to manage several identifiers with one filter only.

25.8.2.2.2 Data reception (transceiver as subscriber)

When LINFlex receives the identifier, the HRF bit in the LINSR is set and, if the HRIE bit in the LINIER is set, an RX interrupt is generated. The software must read the received identifier in the BIDR and specify the data field length using the DFL[0:2] bits in the BDIR before receiving the stop bit of the first byte of data field.

When the checksum reception is completed, an RX interrupt is generated to allow the software to read the received data in the BDR registers.

One or several identifier filters can be configured for reception by resetting the DIR bit in the IFCRx register(s) and activated by setting one or several bits in the IFER.

When at least one identifier filter is configured in reception and activated, and if the received identifier matches the filter, an RX interrupt is generated after the checksum reception only.

Typically, the application has to copy the data from the BDAR to SRAM locations. To copy the data to the right location, the application has to identify the data by means of the identifier. To avoid this and to ease the access to the SRAM locations, the LINFlex controller provides a Filter Match Index. This index value is the number of the filter that matched the received identifier.

The software can use the index in the IFMI register to directly access the pointer that points to the right data array in the SRAM area and copy this data from the BDAR to the SRAM (see [Figure 312](#)).

Using a filter avoids the software reading the ID value in the BIDR, and configuring the direction, the data field length and the checksum type in the BDIR.

If LINFlex cannot provide enough RX identifier filters to handle all identifiers the software has to receive the data for, then a filter can be configured in mask mode (refer to [Section 25.8.2.3, “Slave mode with identifier filtering”](#)) in order to manage several identifiers with one filter only.

25.8.2.2.3 Data discard

When LINFlex receives the identifier, the HRF bit in the LINSR is set and, if the HRIE bit in the LINIER is set, an RX interrupt is generated. If the received identifier does not concern the node, the software must set the DDRQ bit in LINCR2. LINFlex returns to idle state after bit DDRQ is set.

25.8.2.2.4 Error detection

In Slave mode, the following errors are detected:

- **Header error:** An error occurred during header reception (Break Delimiter error, Inconsistent Synch Field, Header Timeout).
- **Bit error:** During transmission, the value read back from the bus differs from the transmitted value.
- **Framing error:** A dominant state has been sampled on the stop bit of the currently received character (synch field, identifier field or data field).
- **Checksum error:** The computed checksum does not match the received one.

25.8.2.2.5 Error handling

In case of Bit Error detection during transmission, LINFlex stops the transmission of the frame after the corrupted bit. LINFlex returns to idle state and an interrupt is generated if the BEIE bit in the LINIER is set.

During reception, a Framing Error leads LINFlex to discard the current frame. LINFlex returns immediately to idle state. An interrupt is generated if the FEIE bit in the LINIER is set.

During reception, a Checksum Error leads LINFlex to discard the received frame. LINFlex returns to idle state. An interrupt is generated if the CEIE bit in the LINIER is set.

During header reception, a Break Delimiter error, an Inconsistent Synch Field or a Timeout error leads LINFlex to discard the header. An interrupt is generated if the HEIE bit in the LINIER is set. LINFlex returns to idle state.

25.8.2.2.6 Valid header

A received header is considered as valid when it has been received correctly according to the LIN protocol.

If a valid Break Field and Break Delimiter come before the end of the current header or at any time during a data field, the current header or data is discarded and the state machine synchronizes on this new break.

25.8.2.2.7 Valid message

A received or transmitted message is considered as valid when the data has been received or transmitted without error according to the LIN protocol.

25.8.2.2.8 Overrun

Once the message buffer is full, the next valid message reception leads to an overrun and a message is lost. The hardware sets the BOF bit in the LINSR to signal the overrun condition. Which message is lost depends on the configuration of the RX message buffer:

- If the buffer lock function is disabled (RBLM bit in LINCR1 cleared) the last message stored in the buffer is overwritten by the new incoming message. In this case the latest message is always available to the application.
- If the buffer lock function is enabled (RBLM bit in LINCR1 set) the most recent message is discarded and the previous message is available in the buffer.

25.8.2.3 Slave mode with identifier filtering

In the LIN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On header reception a slave node decides—depending on the identifier value—whether the software needs to receive or send a response. If the message does not target the node, it must be discarded without software intervention.

To fulfill this requirement, the LINFlex controller provides configurable filters in order to request software intervention only if needed. This hardware filtering saves CPU resources that would otherwise be needed by software for filtering.

25.8.2.3.1 Filter mode

Usually each of the eight IFCR registers filters one dedicated identifier, but this limits the number of identifiers LINFlex can handle to the number of IFCR registers implemented in the device. Therefore, in order to be able to handle more identifiers, the filters can be configured in mask mode.

In **identifier list mode** (the default mode), both filter registers are used as identifier registers. All bits of the incoming identifier must match the bits specified in the filter register.

In **mask mode**, the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”. For the bit mapping and registers organization, please refer to [Figure 311](#).

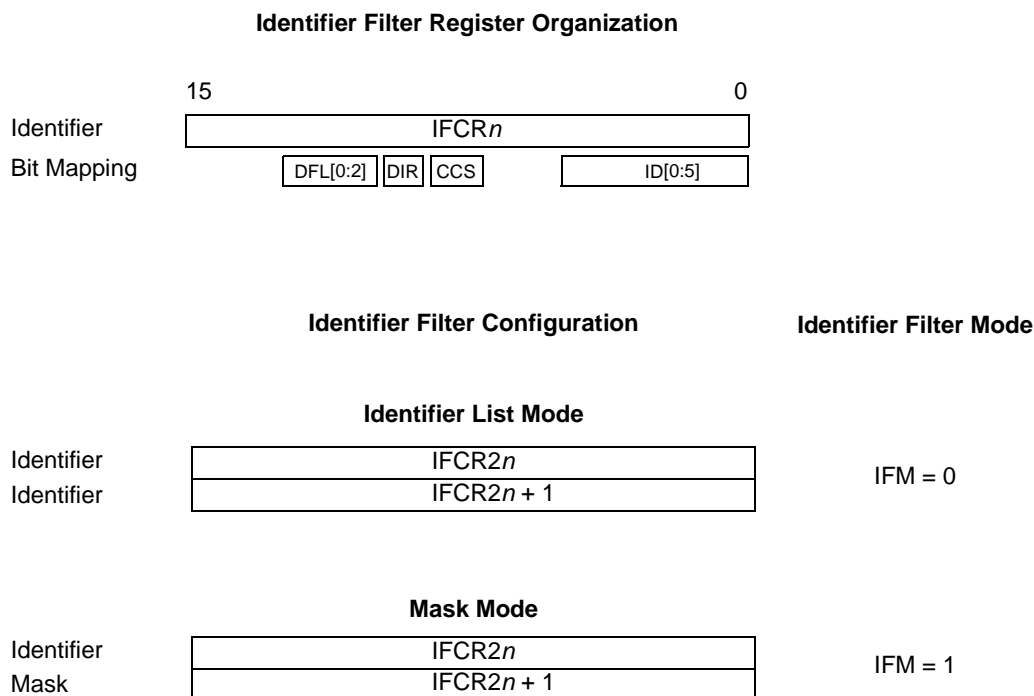


Figure 311. Filter configuration—register organization

25.8.2.3.2 Identifier filter mode configuration

The identifier filters are configured in the IFCR_x registers. To configure an identifier filter the filter must first be deactivated by clearing the FACT bit in the IFER. The **identifier list** or **identifier mask** mode for the corresponding IFCR_x registers is configured by the IFM bit in the IFMR. For each filter, the IFCR_x register configures the ID (or the mask), the direction (TX or RX), the data field length, and the checksum type.

If no filter is active, an RX interrupt is generated on any received identifier event.

If at least one active filter is configured as TX, all received identifiers matching this filter generate a TX interrupt.

If at least one active filter is configured as RX, all received identifiers matching this filter generate an RX interrupt.

If no active filter is configured as RX, all received identifiers not matching TX filter(s) generate an RX interrupt.

Table 342. Filter to interrupt vector correlation

Number of active filters	Number of active filters configured as TX	Number of active filters configured as RX	Interrupt vector
0	0	0	RX interrupt on all identifiers

Table 342. Filter to interrupt vector correlation

Number of active filters	Number of active filters configured as TX	Number of active filters configured as RX	Interrupt vector
a (a > 0)	a	0	— TX interrupt on identifiers matching the filters, — RX interrupt on all other identifiers if BF bit is set, no RX interrupt if BF bit is reset
n (n = a + b)	a (a > 0)	b (b > 0)	— TX interrupt on identifiers matching the TX filters, — RX interrupt on identifiers matching the RX filters, — all other identifiers discarded (no interrupt)
b (b > 0)	0	b	— RX interrupt on identifiers matching the filters, — TX interrupt on all other identifiers if BF bit is set, no TX interrupt if BF bit is reset

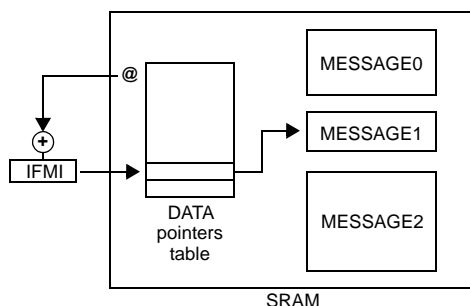


Figure 312. Identifier match index

25.8.2.4 Slave mode with automatic resynchronization

Automatic resynchronization must be enabled in Slave mode if $f_{\text{periph_set_1_clk}}$ tolerance is greater than 1.5%. This feature compensates a $f_{\text{periph_set_1_clk}}$ deviation up to 14%, as specified in LIN standard.

This mode is similar to Slave mode as described in [Section 25.8.2.2, “Slave mode”](#) with the addition of automatic resynchronization enabled by the LASE bit. In this mode LINFlex adjusts the fractional baud rate generator after each Synch Field reception.

Automatic resynchronization method

When automatic resynchronization is enabled, after each LIN Break, the time duration between five falling edges on RDI is sampled on $f_{\text{periph_set_1_clk}}$ and the result of this measurement is stored in an internal 19-bit register called SM (not user accessible) (see [Figure 313](#)). Then the LFDIV value (and its associated registers LINIBRR and LINFBR) are automatically updated at the end of the fifth falling edge. During

LIN Synch Field measurement, the LINFlex state machine is stopped and no data is transferred to the data register.

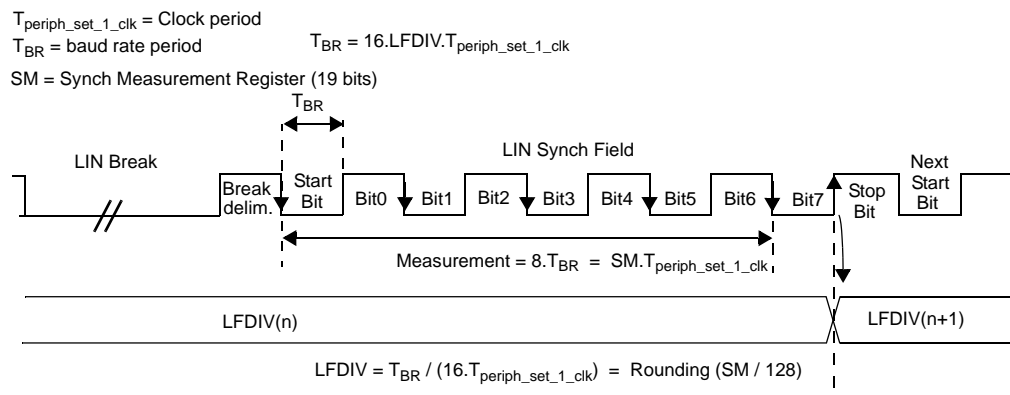


Figure 313. LIN synch field measurement

LFDIV is an unsigned fixed point number. The mantissa is coded on 12 bits in the LINIBRR and the fraction is coded on 4 bits in the LINFBR.

If LASE bit = 1 then LFDIV is automatically updated at the end of each LIN Synch Field.

Three internal registers (not user-accessible) manage the auto-update of the LINFlex divider (LFDIV):

- LFDIV_NOM (nominal value written by software at LINIBRR and LINFBR addresses)
- LFDIV_MEAS (results of the Field Synch measurement)
- LFDIV (used to generate the local baud rate)

On transition to idle, break or break delimiter state due to any error or on reception of a complete frame, hardware reloads LFDIV with LFDIV_NOM.

25.8.2.4.1 Deviation error on the Synch Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Synch Field (relative to the master oscillator). Two checks are performed in parallel.

The first check is based on a measurement between the first falling edge and the last falling edge of the Synch Field:

- If $D1 > 14.84\%$, LHE is set.
- If $D1 < 14.06\%$, LHE is not set.
- If $14.06\% < D1 < 14.84\%$, LHE can be either set or reset depending on the dephasing between the signal on LINFlex_RX pin the $f_{\text{periph_set_1_clk}}$ clock.

The second check is based on a measurement of time between each falling edge of the Synch Field:

- If $D2 > 18.75\%$, LHE is set.
- If $D2 < 15.62\%$, LHE is not set.
- If $15.62\% < D2 < 18.75\%$, LHE can be either set or reset depending on the dephasing between the signal on LINFlex_RX pin the $f_{\text{periph_set_1_clk}}$ clock.

Note that the LINFlex does not need to check if the next edge occurs slower than expected. This is covered by the check for deviation error on the full synch byte.

25.8.2.5 Clock gating

The LINFlex clock can be gated from the Mode Entry module (see). In LIN mode, the LINFlex controller acknowledges a clock gating request once the frame transmission or reception is completed.

25.8.3 8-bit timeout counter

25.8.3.1 LIN timeout mode

Clearing the LTOM bit in the LINTCSR enables the LIN timeout mode. The LINOCCR becomes read-only, and OC1[0:7] and OC2[0:7] output compare values in the LINOCCR are automatically updated by hardware.

This configuration detects header timeout, response timeout, and frame timeout.

Depending on the LIN mode (selected by the MME bit in LINCR1), the 8-bit timeout counter will behave differently.

LIN timeout mode must not be enabled during LIN extended frames transmission or reception (that is, if the data field length in the BIDR is configured with a value higher than 8 data bytes).

25.8.3.1.1 LIN Master mode

Field RTO[0:3] in the LINTOCR can be used to tune response timeout and frame timeout values. Header timeout value is fixed to HTO[0:6] = 28-bit time.

Field OC1[0:7] checks T_{Header} and T_{Response} and field OC2[0:7] checks T_{Frame} (refer to [Figure 314](#)).

When LINFlex moves from Break delimiter state to Synch Field state (refer to [Section 25.7.2.5, “UART mode control register \(UARTCR\)”](#)):

- OC1[0:7] is updated with the value of OC_{Header} ($OC_{\text{Header}} = \text{CNT}[0:7] + 28$),
- OC2[0:7] is updated with the value of OC_{Frame} ($OC_{\text{Frame}} = \text{CNT}[0:7] + 28 + \text{RTO}[0:6] \times 9$ (frame timeout value for an 8-byte frame)),
- the TOCE bit is set.

On the start bit of the first response data byte (and if no error occurred during the header reception), OC1[0:7] is updated with the value of OC_{Response} ($OC_{\text{Response}} = \text{CNT}[0:7] + \text{RTO}[0:6] \times 9$ (response timeout value for an 8-byte frame)).

On the first response byte is received, OC1[0:7] and OC2[0:7] are automatically updated to check T_{Response} and T_{Frame} according to RTO[0:6] (tolerance) and DFL[0:2].

On the checksum reception or in case of error in the header or response, the TOCE bit is reset.

If there is no response, frame timeout value does not take into account the DFL[0:2] value, and an 8-byte response (DFL = 7) is always assumed.

25.8.3.1.2 LIN Slave mode

Field RTO[0:3] in the LINTOCR can be used to tune response timeout and frame timeout values. Header timeout value is fixed to HTO[0:6].

OC1[0:7] checks T_{Header} and T_{Response} and OC2[0:7] checks T_{Frame} (refer to Figure 314).

When LINFlex moves from Break state to Break Delimiter state (refer to Section 25.7.2.5, “UART mode control register (UARTCR)”):

- OC1[0:7] is updated with the value of OC_{Header} ($OC_{\text{Header}} = \text{CNT}[0:7] + \text{HTO}[0:6]$),
- OC2[0:7] is updated with the value of OC_{Frame} ($OC_{\text{Frame}} = \text{CNT}[0:7] + \text{HTO}[0:6] + \text{RTO}[0:6] \times 9$ (frame timeout value for an 8-byte frame)),
- The TOCE bit is set.

On the start bit of the first response data byte (and if no error occurred during the header reception), OC1[0:7] is updated with the value of OC_{Response} ($OC_{\text{Response}} = \text{CNT}[0:7] + \text{RTO}[0:7] \times 9$ (response timeout value for an 8-byte frame)).

Once the first response byte is received, OC1[0:7] and OC2[0:7] are automatically updated to check T_{Response} and T_{Frame} according to RTO[0:6] (tolerance) and DFL[0:2].

On the checksum reception or in case of error in the header or data field, the TOCE bit is reset.

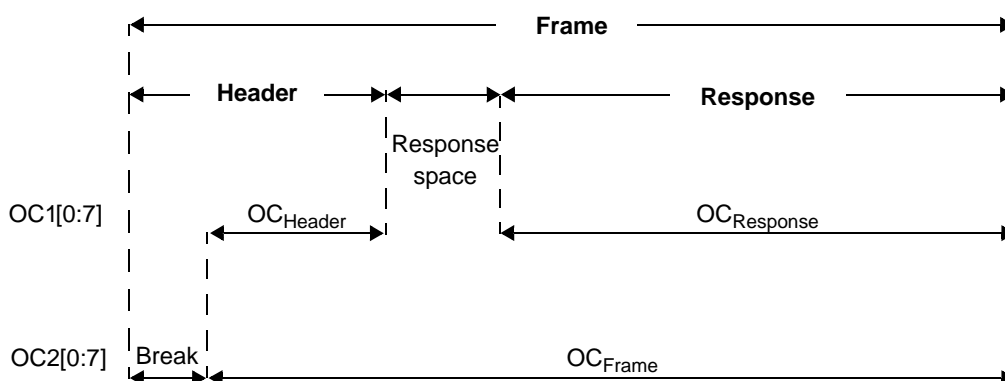


Figure 314. Header and response timeout

25.8.3.2 Output compare mode

Setting the LTOM bit in the LINTCSR enables the output compare mode. This mode allows the user to fully customize the use of the counter.

OC1[0:7] and OC2[0:7] output compare values can be updated in the LINTOCR by software.

25.8.4 Interrupts

Table 343. LINFlex interrupt control

Interrupt event	Event flag bit	Enable control bit	Interrupt vector
Header Received interrupt	HRF	HRIE	RXI ¹
Data Transmitted interrupt	DTF	DTIE	TXI
Data Received interrupt	DRF	DRIE	RXI
Data Buffer Empty interrupt	DBEF	DBEIE	TXI
Data Buffer Full interrupt	DBFF	DBFIE	RXI
Wake-up interrupt	WUPF	WUPIE	RXI
LIN State interrupt ²	LSF	LSIE	RXI
Buffer Overrun interrupt	BOF	BOIE	ERR
Framing Error interrupt	FEF	FEIE	ERR
Header Error interrupt	HEF	HEIE	ERR
Checksum Error interrupt	CEF	CEIE	ERR
Bit Error interrupt	BEF	BEIE	ERR
Output Compare interrupt	OCF	OCIE	ERR
Stuck at Zero interrupt	SZF	SZIE	ERR

¹ In Slave mode, if at least one filter is configured as TX and enabled, header received interrupt vector is RXI or TXI depending on the value of identifier received.

² For debug and validation purposes

Chapter 26

FlexCAN Module

26.1 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. A general block diagram is shown in [Figure 315](#), which describes the main sub-blocks implemented in the FlexCAN module, including two embedded memories, one for storing Message Buffers (MB) and another one for storing Rx Individual Mask Registers. Support for up to 32 Message Buffers is provided.

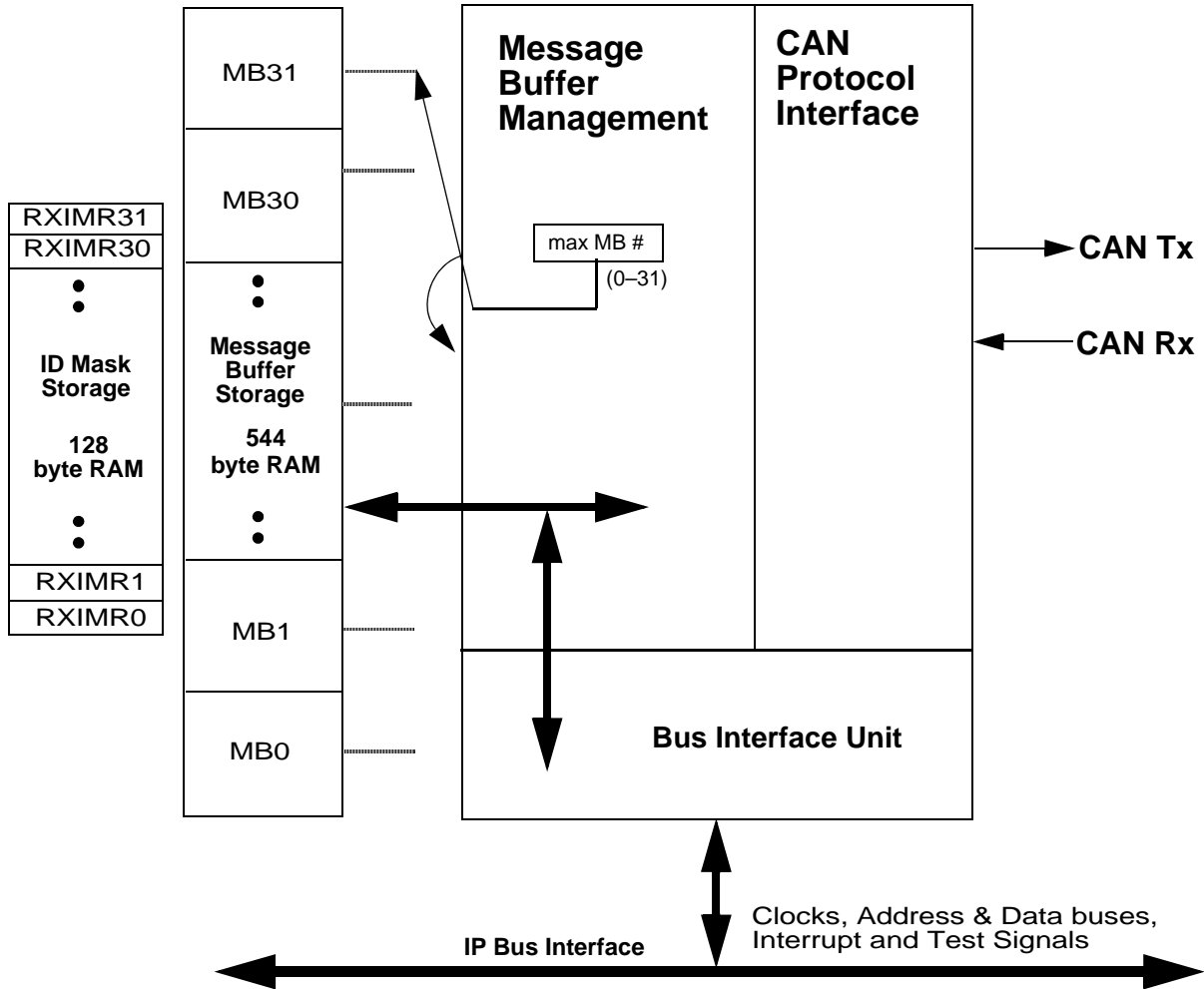


Figure 315. FlexCAN Block Diagram

26.1.1 Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B [Ref. 1], which supports both standard and extended message frames. A flexible number of Message Buffers (32) is also supported. The Message Buffers are stored in an embedded RAM dedicated to the FlexCAN module.

The CAN Protocol Interface (CPI) sub-module manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages and

performing error handling. The Message Buffer Management (MBM) sub-module handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the Bus Interface Unit.

26.1.2 FlexCAN Module Features

The FlexCAN module includes these distinctive features:

- Full Implementation of the CAN protocol specification, Version 2.0B
 - Standard data and remote frames
 - Extended data and remote frames
 - Zero to eight bytes data length
 - Programmable bit rate up to 1 Mbit/s
 - Content-related addressing
- Flexible Message Buffers (up to 32) of zero to eight bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Message Buffer
- Includes either 544 bytes (32 MBs) of RAM used for MB storage
- Includes either 128 bytes (32 MBs) of RAM used for individual Rx Mask Registers
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard or 32 partial (8 bits) IDs, with individual masking capability
- Selectable backwards compatibility with previous FlexCAN version
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused MB and Rx Mask Register space can be used as general purpose RAM space
- Listen only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity

26.1.3 Modes of Operation

The FlexCAN module has four functional modes: Normal Mode (User and Supervisor), Freeze Mode, Listen-Only Mode and Loop-Back Mode. There are also three low power modes: Disable Mode and Stop Mode.

- Normal Mode (User or Supervisor):

In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze Mode:

It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested at MCU level. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See [Section 26.4.9.1, “Freeze Mode”](#), for more information.

- Listen-Only Mode:

The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back Mode:

The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable Mode:

This low power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU. When disabled, the module requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. Exit from this mode is done by negating the MDIS bit in the MCR Register. See [Section 26.4.9.2, “Module Disable Mode”](#), for more information.

- Stop Mode:

This low power mode is entered when Stop Mode is requested at MCU level. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop Mode request is removed or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See [Section 26.4.9.3, “Stop Mode”](#), for more information.

26.2 External Signal Description

26.2.1 Overview

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in [Table 344](#) and described in more detail in the next subsections.

Table 344. FlexCAN Signals

Signal Name ¹	Direction	Description
CAN Rx	Input	CAN Receive Pin
CAN Tx	Output	CAN Transmit Pin

¹ The actual MCU pins may have different names. Please consult the Device User Guide for the actual signal names.

26.2.2 Signal Descriptions

26.2.2.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.

26.2.2.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.

26.3 Memory Map/Register Definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the MCU. The addresses presented here are relative to the base address.

The address space occupied by FlexCAN has 96 bytes for registers starting at the module base address, followed by MB storage space in embedded RAM starting at address 0x0060, and an extra ID Mask storage space in a separate embedded RAM starting at address 0x0880.

26.3.1 FlexCAN Memory Mapping

The complete memory map for a FlexCAN module with 32 MBs capability is shown in [Table 345](#). Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV bit in the MCR Register. These registers are identified as S/U in the Access column of [Table 345](#).

The IFLAG2 and IMASK2 registers are considered reserved space when FlexCAN is configured with 16 or 32 MBs. The Rx Global Mask (RXGMASK), Rx Buffer 14 Mask (RX14MASK) and the Rx Buffer 15

Mask (RX15MASK) registers are provided for backwards compatibility, and are not used when the BCC bit in MCR is asserted.

The memory sizes are 544 and 128 bytes, so the address ranges 0x0280–0x047F and 0x0900–0x097F are considered reserved space. Furthermore, if the BCC bit in MCR is negated, then the whole Rx Individual Mask Registers address range (0x0880–0x097F) is considered reserved space.

Table 345. Module Memory Map

Address	Use	Access Type	Affected by Hard Reset	Affected by Soft Reset
Base + 0x0000	Module Configuration (MCR)	S	Yes	Yes
Base + 0x0004	Control Register (CTRL)	S/U	Yes	No
Base + 0x0008	Free Running Timer (TIMER)	S/U	Yes	Yes
Base + 0x000C	Reserved			
Base + 0x0010	Rx Global Mask (RXGMASK)	S/U	Yes	No
Base + 0x0014	Rx Buffer 14 Mask (RX14MASK)	S/U	Yes	No
Base + 0x0018	Rx Buffer 15 Mask (RX15MASK)	S/U	Yes	No
Base + 0x001C	Error Counter Register (ECR)	S/U	Yes	Yes
Base + 0x0020	Error and Status Register (ESR)	S/U	Yes	Yes
Base + 0x0028	Interrupt Masks 1 (IMASK1)	S/U	Yes	Yes
Base + 0x0030	Interrupt Flags 1 (IFLAG1)	S/U	Yes	Yes
Base + 0x0034–0x007F	Reserved			
Base + 0x0080–0x017F	Message Buffers MB0–MB15	S/U	No	No
Base + 0x0180–087F	Reserved			
Base + 0x0880–0x08BF	Rx Individual Mask Registers RXIMR0–RXIMR15	S/U	No	No
Base + 0x08C0–0x08FF	Reserved			

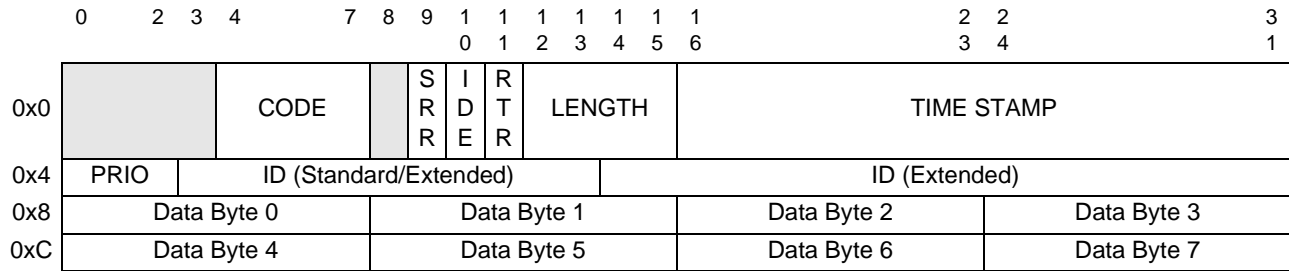
The FlexCAN module stores CAN messages for transmission and reception using a Message Buffer structure. Each individual MB is formed by 16 bytes mapped on memory as described in [Table 346](#). [Table 346](#) shows a Standard/Extended Message Buffer (MB0) memory map, using 16 bytes total (0x80–0x8F space).

Table 346. Message Buffer MB0 Memory Mapping

Address Offset	MB Field
0x80	Control and Status (C/S)
0x84	Identifier Field
0x88–0x8F	Data Field 0 – Data Field 7 (1 byte each)

26.3.2 Message Buffer Structure

The Message Buffer structure used by the FlexCAN module is represented in [Figure 316](#). Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification (Version 2.0 Part B) are represented.




 = Unimplemented or Reserved

Figure 316. Message Buffer Structure

CODE — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the Flexcan module itself, as part of the message buffer matching and arbitration process. The encoding is shown in [Table 347](#) and [Table 348](#). See [Section 26.4, “Functional Description”](#), for additional information.

Table 347. Message Buffer Code for Rx buffers

Rx Code BEFORE Rx New Frame	Description	Rx Code AFTER Rx New Frame	Comment
0000	INACTIVE: MB is not active.	–	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.
0010	FULL: MB is full.	0010	The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to Section 26.4.5, “Matching Process” , for details about overrun behavior.

Table 347. Message Buffer Code for Rx buffers

Rx Code BEFORE Rx New Frame	Description	Rx Code AFTER Rx New Frame	Comment
0110	OVERRUN: a frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Section 26.4.5, "Matching Process" , for details about overrun behavior.
0XY1 ¹	BUSY: Flexcan is updating the contents of the MB. The CPU must not access the MB.	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

¹ Note that for Tx MBs (see [Table 348](#)), the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register.

Table 348. Message Buffer Code for Tx buffers

RTR	Initial Tx code	Code after successful transmission	Description
X	1000	–	INACTIVE: MB does not participate in the arbitration process.
0	1100	1000	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1	1100	0100	Transmit remote frame unconditionally once. After transmission, the MB automatically becomes an Rx MB with the same ID.
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the Code field is automatically updated to '1110' to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the Code automatically returns to '1010' to restart the process again.
0	1110	1010	This is an intermediate code that is automatically written to the MB by the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to '1010'. The CPU can also write this code with the same effect.

SRR — Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss.

- 1 = Recessive value is compulsory for transmission in Extended Format frames
- 0 = Dominant is not a valid value for transmission in Extended Format frames

IDE — ID Extended Bit

This bit identifies whether the frame format is standard or extended.

- 1 = Frame format is extended
- 0 = Frame format is standard

RTR — Remote Transmission Request

This bit is used for requesting transmissions of a data frame. If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission.

- 1 = Indicates the current MB has a Remote Frame to be transmitted
- 0 = Indicates the current MB has a Data Frame to be transmitted

NOTE

Do not configure the last Message Buffer to be RTR frame.

LENGTH — Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see [Figure 316](#)). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field.

TIME STAMP — Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO — Local priority

This 3-bit field is only used when LPRIO_EN bit is set in MCR and it only makes sense for Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See [Section 26.4.3, "Arbitration process"](#).

ID — Frame Identifier

In Standard Frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.

DATA — Data Field

Up to eight bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame.

26.3.3 Rx FIFO Structure

When the FEN bit is set in the MCR, the memory area from 0x80 to 0xFC (which is normally occupied by MBs 0 to 7) is used by the reception FIFO engine. [Figure 317](#) shows the Rx FIFO data structure. The region 0x80-0x8C contains an MB structure which is the port through which the CPU reads data from the FIFO (the oldest frame received and not read yet). The region 0x90-0xDC is reserved for internal use of the FIFO engine. The region 0xE0-0xFC contains an 8-entry ID table that specifies filtering criteria for accepting frames into the FIFO. [Figure 318](#) shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the MCR. Note that all elements of the table must have the same format. See [Section 26.4.7, “Rx FIFO”](#), for more information.

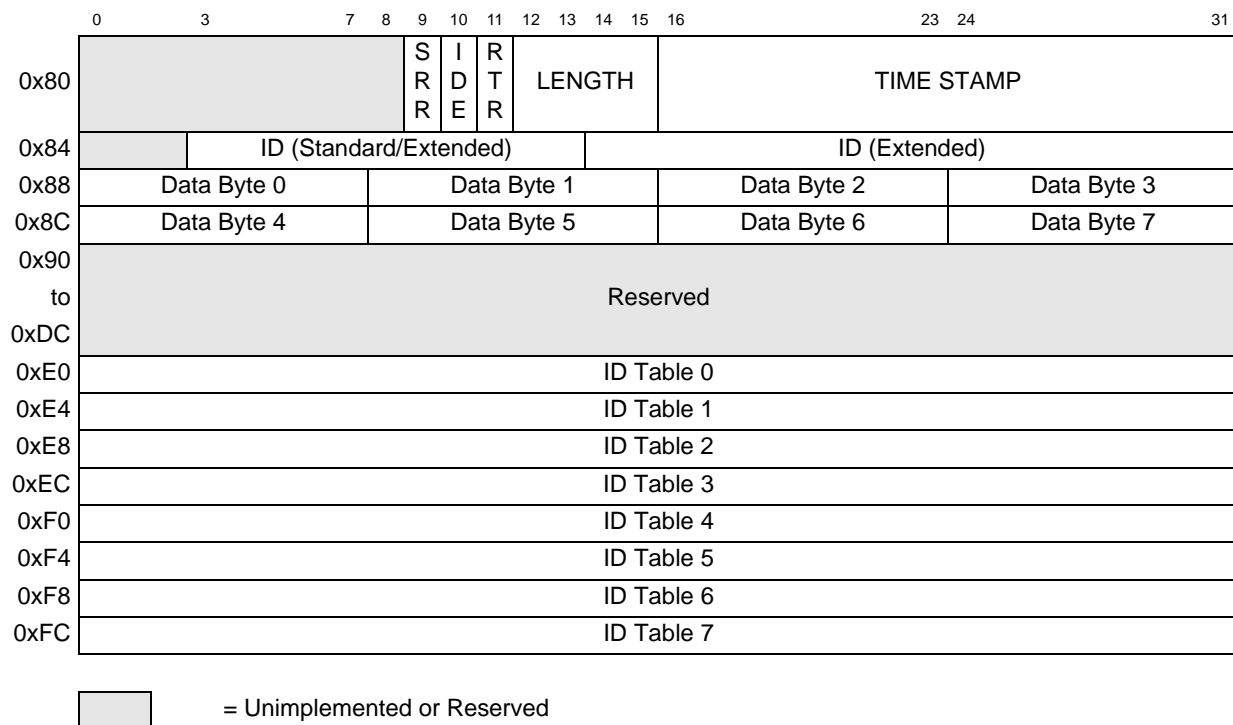
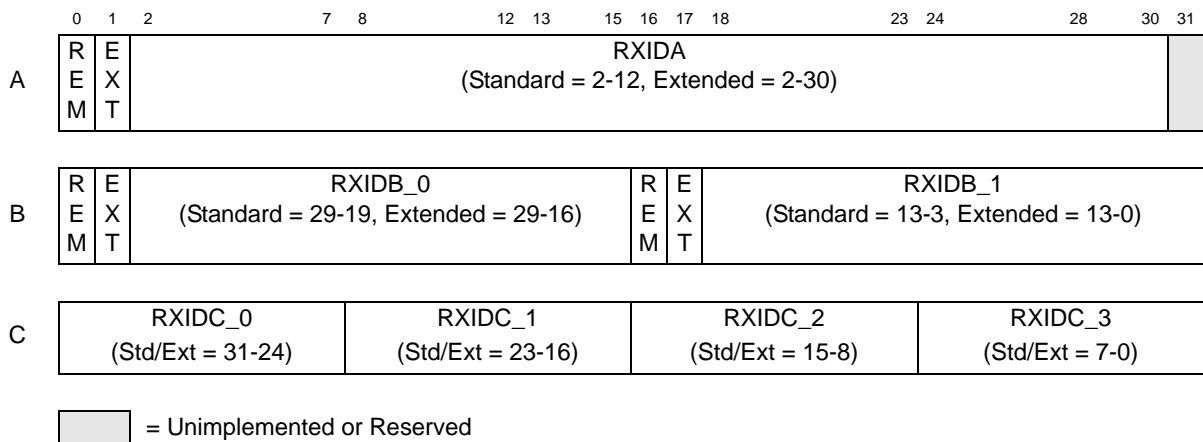


Figure 317. Rx FIFO Structure


Figure 318. ID Table 0 - 7

REM — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

EXT — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12 and 18 to 28) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

26.3.4 Register Descriptions

The FlexCAN registers are described in this section in ascending address order.

26.3.4.1 Module Configuration Register (MCR)

This register defines global system configurations, such as the module operation mode (e.g., low power) and maximum message buffer configuration. This register can be accessed at any time, however some fields must be changed only during Freeze Mode. Find more information in the fields descriptions ahead.

		Base + 0x0000															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		MDIS	FRZ	FEN	HALT	NOT_RDY	WAK_MSK	SOFT_RST	FRZ_ACK	SUPV	SLF_WAK	WRN_EN	LPM_ACK	WAK_SRC	0	SRX_DIS	BCC
W																	
RESET:		Note ₁	1	0	1	1	0	0	Note ₃	1	0	0	Note ₄	0	0	0	0

		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0	0	LPRI_O_EN	0 ²	0	0	IDAM	0	0	MAXMB						
W																	
RESET:		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

= Unimplemented or Reserved

- 1 Reset value of this bit is different on various platforms. Consult the specific MCU documentation to determine its value.
- 2 This bit must always be written 0.
- 3 Different on various platforms, but it is always the opposite of the MDIS reset value.
- 4 Different on various platforms, but it is always the same as the MDIS reset value.

Figure 319. Module Configuration Register (MCR)

MDIS — Module Disable

This bit controls whether FlexCAN is enabled or not. When disabled, the module requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. This is the only bit in MCR not affected by soft reset. See [Section 26.4.9.2, “Module Disable Mode”](#), for more information.

- 1 = Disable the FlexCAN module
- 0 = Enable the FlexCAN module

FRZ — Freeze Enable

The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at MCU level. When FRZ is asserted, FlexCAN is enabled to enter Freeze Mode. Negation of this bit field causes FlexCAN to exit from Freeze Mode.

- 1 = Enabled to enter Freeze Mode
- 0 = Not enabled to enter Freeze Mode

FEN — FIFO Enable

This bit controls whether the FIFO feature is enabled or not. When FEN is set, MBs 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xFF) is used by the FIFO engine. See [Section 26.3.3, “Rx FIFO Structure”](#) and [Section 26.4.7, “Rx FIFO”](#), for more information. This bit must be written in Freeze mode only.

- 1 = FIFO enabled
- 0 = FIFO not enabled

HALT — Halt FlexCAN

Assertion of this bit puts the FlexCAN module into Freeze Mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in Freeze Mode, the CPU has write access to the Error Counter Register, that is otherwise read-only. Freeze Mode can not be entered while FlexCAN is in any of the low power modes. See [Section 26.4.9.1, “Freeze Mode”](#), for more information.

- 1 = Enters Freeze Mode if the FRZ bit is asserted.
- 0 = No Freeze Mode request.

NOT_RDY — FlexCAN Not Ready

This read-only bit indicates that FlexCAN is either in Disable Mode, Stop Mode, or Freeze Mode. It is negated once FlexCAN has exited these modes.

- 1 = FlexCAN module is either in Disable Mode, Stop Mode, or Freeze Mode
- 0 = FlexCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode

WAK_MSK — Wake Up Interrupt Mask

This bit enables the Wake Up Interrupt generation.

- 1 = Wake Up Interrupt is enabled
- 0 = Wake Up Interrupt is disabled

SOFT_RST — Soft Reset

When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMASK1, IMASK2, IFLAG1, IFLAG2. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected:

- CTRL
- RXIMR0–RXIMR63
- RXGMASK, RX14MASK, RX15MASK
- all Message Buffers

The SOFT_RST bit can be asserted directly by the CPU when it writes to the MCR Register, but it is also asserted when global soft reset is requested at MCU level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed.

Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied.

- 1 = Resets the registers marked as “affected by soft reset” in [Table 345](#)
- 0 = No reset request

FRZ_ACK — Freeze Mode Acknowledge

This read-only bit indicates that FlexCAN is in Freeze Mode and its prescaler is stopped. The Freeze Mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered Freeze Mode. If Freeze Mode request is negated, then this bit is negated once the FlexCAN prescaler is running again. If Freeze Mode is requested while FlexCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See [Section 26.4.9.1, “Freeze Mode”](#), for more information.

- 1 = FlexCAN in Freeze Mode, prescaler stopped
- 0 = FlexCAN not in Freeze Mode, prescaler running

SUPV — Supervisor Mode

This bit configures some of the FlexCAN registers to be either in Supervisor or Unrestricted memory space. The registers affected by this bit are marked as S/U in the Access Type column of [Table 345](#). Reset value of this bit is ‘1’, so the affected registers start with Supervisor access restrictions. This bit should be written in Freeze mode only.

- 1 = Affected registers are in Supervisor memory space. Any access without supervisor permission behaves as though the access was done to an unimplemented register location
- 0 = Affected registers are in Unrestricted memory space

SLF_WAK — Self Wake Up

This bit enables the Self Wake Up feature when FlexCAN is in Stop Mode. If this bit had been asserted by the time FlexCAN entered Stop Mode, then FlexCAN will look for a recessive to dominant transition on the bus during these modes. If a transition from recessive to dominant is detected during Stop Mode, then FlexCAN generates, if enabled to do so, a Wake Up interrupt to the CPU so that it can resume the clocks globally. This bit can not be written while the module is in Stop Mode.

- 1 = FlexCAN Self Wake Up feature is enabled
- 0 = FlexCAN Self Wake Up feature is disabled

WRN_EN — Warning Interrupt Enable

When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit must be written in Freeze mode only.

- 1 = TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96 .
- 0 = TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters.

LPM_ACK — Low Power Mode Acknowledge

This read-only bit indicates that FlexCAN is either in Disable Mode or Stop Mode. Either of these low power modes can not be entered until all current transmission or reception processes have finished, so the CPU can poll the LPM_ACK bit to know when FlexCAN has actually entered low power mode. See [Section 26.4.9.2, “Module Disable Mode”](#) and [Section 26.4.9.3, “Stop Mode”](#), for more information.

- 1 = FlexCAN is either in Disable Mode or Stop mode
- 0 = FlexCAN not in any of the low power modes

WAK_SRC — Wake Up Source

This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake up. See [Section 26.4.9.3, “Stop Mode”](#), for more information. This bit should be written in Freeze mode only.

- 1 = FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus
- 0 = FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus.

NOTE

The integrated low-pass filter may not be available in all MCUs. In case it is not available, the unfiltered input is always used for wake up purposes, and this bit has no effect on the FlexCAN operation.

SRX_DIS — Self Reception Disable

This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit must be written in Freeze mode only.

- 1 = Self reception disabled
- 0 = Self reception enabled

BCC — Backwards Compatibility Configuration

This bit is provided to support Backwards Compatibility with previous FlexCAN versions. When this bit is negated, the following configuration is applied:

- For MCUs supporting individual Rx ID masking, this feature is disabled. Instead of individual ID masking per MB, FlexCAN uses its previous masking scheme with RXGMASK, RX14MASK and RX15MASK.
- The reception queue feature is disabled. Upon receiving a message, if the first MB with a matching ID that is found is still occupied by a previous unread message, FlexCAN will not look for another matching MB. It will override this MB with the new message and set the CODE field to ‘0110’ (overrun).

Upon reset this bit is negated, allowing legacy software to work without modification. This bit must be written in Freeze mode only.

- 1 = Individual Rx masking and queue feature are enabled.
- 0 = Individual Rx masking and queue feature are disabled.

LPRIO_EN— Local Priority Enable

This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit must be written in Freeze mode only.

- 1 = Local Priority enabled
- 0 = Local Priority disabled

IDAM — ID Acceptance Mode

This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown in [Table 349](#). Note that all elements of the table are configured at the same time by this field (they are all the same format). See [Section 26.3.3, “Rx FIFO Structure”](#). This bit must be written in Freeze mode only.

Table 349. IDAM Coding

IDAM	Format	Explanation
0b00	A	One full ID (standard or extended) per filter element.
0b01	B	Two full standard IDs or two partial 14-bit extended IDs per filter element.
0b10	C	Four partial 8-bit IDs (standard or extended) per filter element.
0b11	D	All frames rejected.

MAXMB — Maximum Number of Message Buffers

This 6-bit field defines the maximum number of message buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field must be changed only while the module is in Freeze Mode.

Maximum MBs in use = MAXMB + 1.

NOTE

MAXMB must be programmed with a value smaller or equal to the number of available Message Buffers, otherwise FlexCAN can transmit and receive wrong messages.

26.3.4.2 Control Register (CTRL)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler. This register can be accessed at any time, however some fields must be changed only during either Disable Mode or Freeze Mode. Find more information in the fields descriptions ahead.

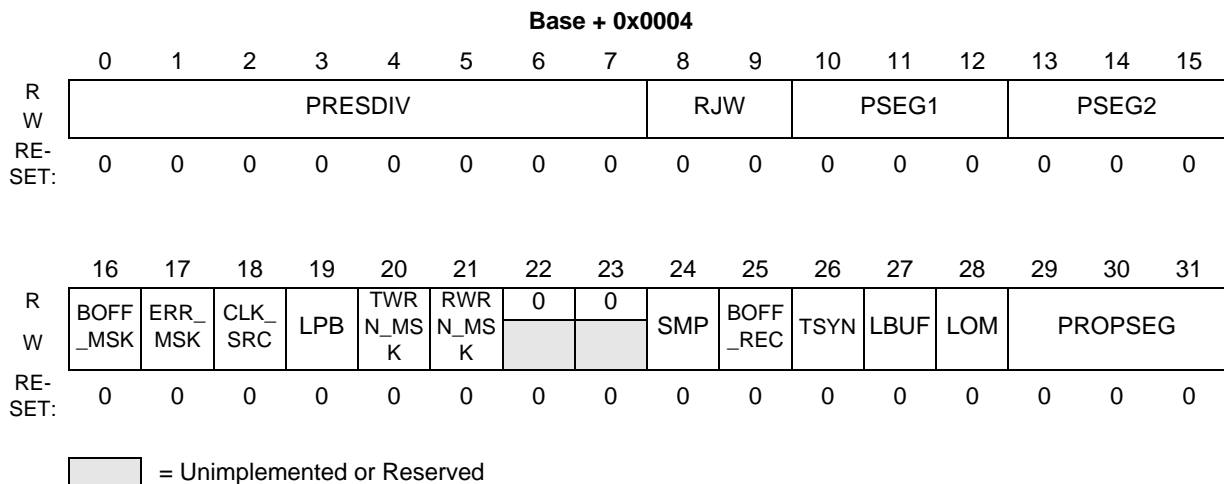


Figure 320. Control Register (CTRL)

PRESDIV — Prescaler Division Factor

This 8-bit field defines the ratio between the CPI clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the CPI clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the CPI clock frequency divided by 256. For more information refer to [Section 26.4.8.4, “Protocol Timing”](#).¹ This bit must be written in Freeze mode only.

$$\text{Sclock frequency} = \text{CPI clock frequency} / (\text{PRESDIV} + 1)$$

RJW — Resync Jump Width

This 2-bit field defines the maximum number of time quanta¹ that a bit time can be changed by one re-synchronization. The valid programmable values are 0–3. This bit must be written in Freeze mode only.

$$\text{Resync Jump Width} = \text{RJW} + 1.$$

PSEG1 — Phase Segment 1

This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. This bit must be written in Freeze mode only.

$$\text{Phase Buffer Segment 1} = (\text{PSEG1} + 1) \times \text{Time-Quanta}.$$

PSEG2 — Phase Segment 2

This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. This bit must be written in Freeze mode only.

$$\text{Phase Buffer Segment 2} = (\text{PSEG2} + 1) \times \text{Time-Quanta}.$$

BOFF_MSK — Bus Off Mask

This bit provides a mask for the Bus Off Interrupt.

- One time quantum is equal to the Sclock period.

FlexCAN Module

- 1 = Bus Off interrupt enabled
- 0 = Bus Off interrupt disabled

ERR_MSK — Error Mask

This bit provides a mask for the Error Interrupt.

- 1 = Error interrupt enabled
- 0 = Error interrupt disabled

CLK_SRC — CAN Engine Clock Source

This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Scklock). In order to guarantee reliable operation, this bit must only be changed while the module is in Disable Mode. See [Section 26.4.8.4, “Protocol Timing”](#), for more information.

- 1 = The CAN engine clock source is the bus clock
- 0 = The CAN engine clock source is the oscillator clock

NOTE

This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL clock feeding the FlexCAN module. In these cases, this bit has no effect on the module operation.

TWRN_MSK — Tx Warning Interrupt Mask

This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated.

- 1 = Tx Warning Interrupt enabled
- 0 = Tx Warning Interrupt disabled

RWRN_MSK — Rx Warning Interrupt Mask

This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated.

- 1 = Rx Warning Interrupt enabled
- 0 = Rx Warning Interrupt disabled

LPB — Loop Back

This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN

ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit must be written in Freeze mode only.

- 1 = Loop Back enabled
- 0 = Loop Back disabled

SMP — Sampling Mode

This bit defines the sampling mode of CAN bits at the Rx input. This bit must be written in Freeze mode only.

- 1 = Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used
- 0 = Just one sample is used to determine the bit value

BOFF_REC — Bus Off Recovery Mode

This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery.

- 1 = Automatic recovering from Bus Off state disabled
- 0 = Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B

TSYN — Timer Sync Mode

This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message (i.e., global network time). If the FEN bit in MCR is set (FIFO enabled), MB8 is used for timer synchronization instead of MB0. This bit must be written in Freeze mode only.

- 1 = Timer Sync feature enabled
- 0 = Timer Sync feature disabled

LBUF — Lowest Buffer Transmitted First

This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration. This bit must be written in Freeze mode only.

- 1 = Lowest number buffer is transmitted first
- 0 = Buffer with highest priority is transmitted first

LOM — Listen-Only Mode

This bit configures FlexCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. This bit must be written in Freeze mode only.

1 = FlexCAN module operates in Listen Only Mode

0 = Listen Only Mode is deactivated

PROPSEG — Propagation Segment

This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This bit must be written in Freeze mode only.

Propagation Segment Time = (PROPSEG + 1) * Time-Quanta.

Time-Quantum = one Sclock period.

26.3.4.3 Free Running Timer (TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0000 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During Freeze Mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

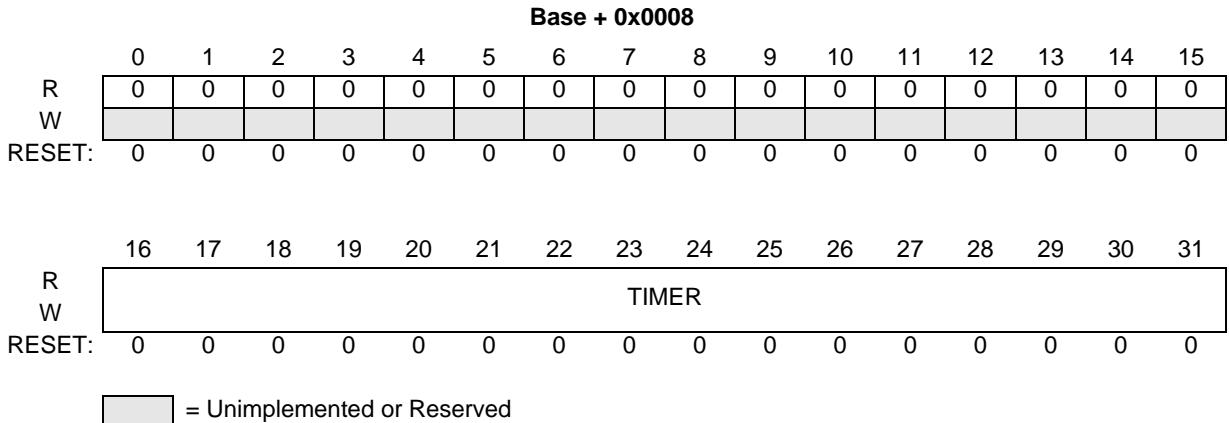


Figure 321. Free Running Timer (TIMER)

26.3.4.4 Rx Global Mask (RXGMASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RXGMASK Register to have no effect on the module operation. For MCUs not supporting individual masks per MB, this register is always effective.

RXGMASK is used as acceptance mask for all Rx MBs, excluding MBs 14–15, which have individual mask registers. When the FEN bit in MCR is set (FIFO enabled), the RXGMASK also applies to all elements of the ID filter table, except elements 6-7, which have individual masks.

Refer to [Section 26.4.7, "Rx FIFO"](#) for important details on usage of RXGMASK on filtering process for Rx FIFO.

The contents of this register must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

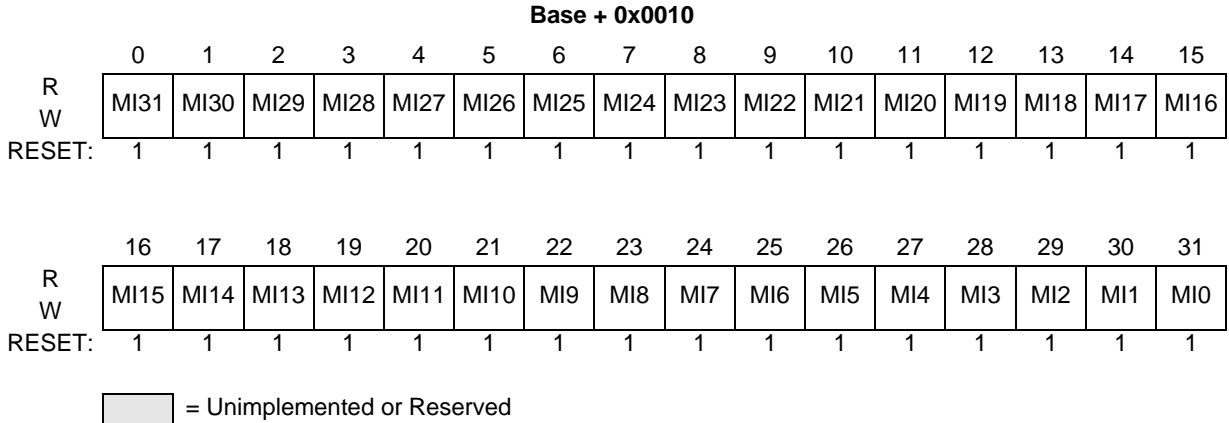


Figure 322. Rx Global Mask Register (RXGMASK)

MI31–MI0 — Mask Bits

For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR).

- 1 = The corresponding bit in the filter is checked against the one received
- 0 = The corresponding bit in the filter is “don’t care”

26.3.4.5 Rx 14 Mask (RX14MASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RX14MASK Register to have no effect on the module operation.

RX14MASK is used as acceptance mask for the Identifier in Message Buffer 14. When the FEN bit in MCR is set (FIFO enabled), the RXG14MASK also applies to element 6 of the ID filter table. This register has the same structure as the Rx Global Mask Register.

Refer to [Section 26.4.7, "Rx FIFO"](#) for important details on usage of RX14MASK on filtering process for Rx FIFO.

The contents of this register must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

- Address Offset: 0x14
- Reset Value: 0xFFFF_FFFF

26.3.4.6 Rx 15 Mask (RX15MASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RX15MASK Register to have no effect on the module operation.

When the BCC bit is negated, RX15MASK is used as acceptance mask for the Identifier in Message Buffer 15. When the FEN bit in MCR is set (FIFO enabled), the RXG15MASK also applies to element 7 of the ID filter table. This register has the same structure as the Rx Global Mask Register.

Refer to [Section 26.4.7, "Rx FIFO"](#) for important details on usage of RX15MASK on filtering process for Rx FIFO.

The contents of this register must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

- Address Offset: 0x18
- Reset Value: 0xFFFF_FFFF

26.3.4.7 Error Counter Register (ECR)

This register has two 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (Tx_Err_Counter field) and Receive Error Counter (Rx_Err_Counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely

implemented in the FlexCAN module. Both counters are read only except in Freeze Mode, where they can be written by the CPU.

Writing to the Error Counter Register while in Freeze Mode is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol, e.g. transmit 'Error Active' or 'Error Passive' flag, delay its transmission start time ('Error Passive') and avoid any influence on the bus when in 'Bus Off' state. The following are the basic rules for FlexCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Passive' state.
- If the FlexCAN state is 'Error Passive', and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Active' state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect 'Bus Off' state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to zero.
- If FlexCAN is in 'Bus Off' state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be 'Error Active' and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to 'Error Passive' state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore the device never goes to the 'Bus Off' state.
- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to 'Error Active' state.

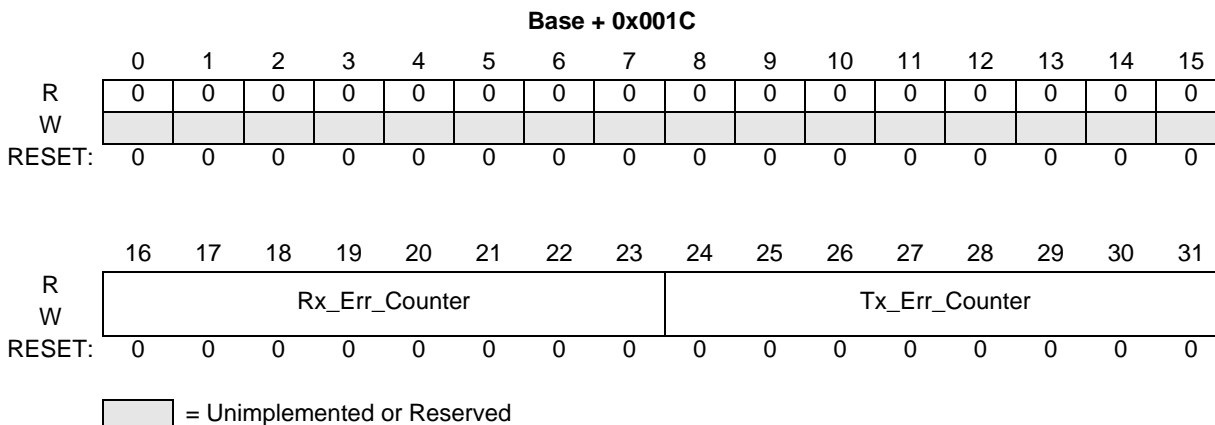


Figure 323. Error Counter Register (ECR)

26.3.4.8 Error and Status Register (ESR)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the CPU. The reported error conditions (bits 16-21) are those that occurred since the last time the CPU read this register. The CPU read action clears bits 16-21. Bits 22-28 are status bits.

Most bits in this register are read only, except TWRN_INT, RWRN_INT, BOFF_INT, WAK_INT and ERR_INT, that are interrupt flags that can be cleared by writing ‘1’ to them (writing ‘0’ has no effect). See [Section 26.4.10, “Interrupts”](#), for more details.

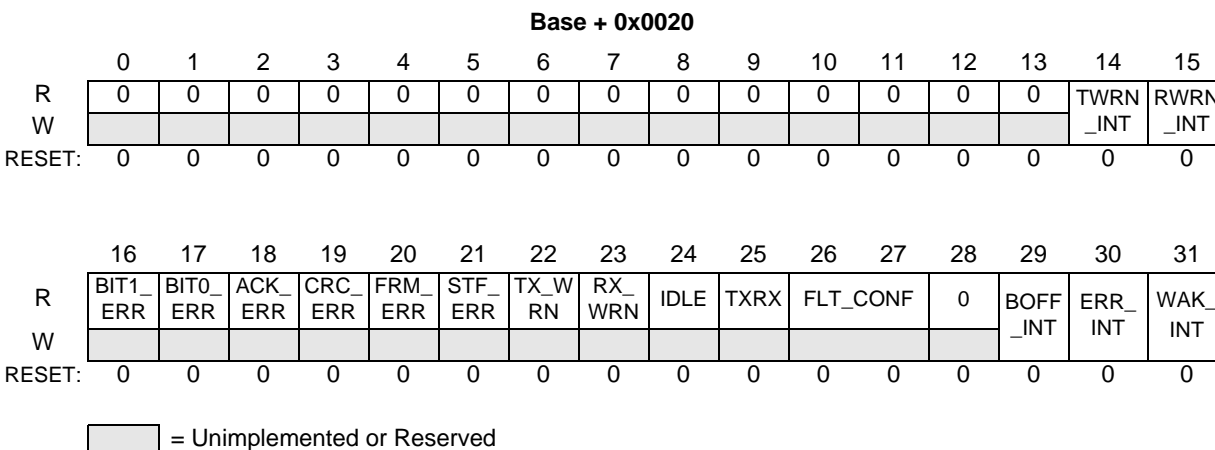


Figure 324. Error and Status Register (ESR)

TWRN_INT — Tx Warning Interrupt Flag

If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from ‘0’ to ‘1’, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect.

- 1 = The Tx error counter transition from < 96 to ≥ 96
- 0 = No such occurrence

RWRN_INT — Rx Warning Interrupt Flag

If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from '0' to '1', meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to '1'. Writing '0' has no effect.

- 1 = The Rx error counter transition from < 96 to ≥ 96
- 0 = No such occurrence

BIT1_ERR — Bit1 Error

This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.

- 1 = At least one bit sent as recessive is received as dominant
- 0 = No such occurrence

NOTE

This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.

BIT0_ERR — Bit0 Error

This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.

- 1 = At least one bit sent as dominant is received as recessive
- 0 = No such occurrence

ACK_ERR — Acknowledge Error

This bit indicates that an Acknowledge Error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT.

- 1 = An ACK error occurred since last read of this register
- 0 = No such occurrence

CRC_ERR — Cyclic Redundancy Check Error

This bit indicates that a CRC Error has been detected by the receiver node, i.e., the calculated CRC is different from the received.

- 1 = A CRC error occurred since last read of this register.
- 0 = No such occurrence

FRM_ERR — Form Error

This bit indicates that a Form Error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit.

- 1 = A Form Error occurred since last read of this register
- 0 = No such occurrence

STF_ERR — Stuffing Error

This bit indicates that a Stuffing Error has been detected.

FlexCAN Module

1 = A Stuffing Error occurred since last read of this register.
 0 = No such occurrence.

TX_WRN — TX Error Warning

This bit indicates when repetitive errors are occurring during message transmission.

1 = TX_Err_Counter \geq 96
 0 = No such occurrence

RX_WRN — Rx Error Warning

This bit indicates when repetitive errors are occurring during message reception.

1 = Rx_Err_Counter \geq 96
 0 = No such occurrence

IDLE — CAN bus IDLE state

This bit indicates when CAN bus is in IDLE state.

1 = CAN bus is now IDLE
 0 = No such occurrence

TXRX — Current FlexCAN status (transmitting/receiving)

This bit indicates if FlexCAN is transmitting or receiving a message when the CAN bus is not in IDLE state. This bit has no meaning when IDLE is asserted.

1 = FlexCAN is transmitting a message (IDLE=0)
 0 = FlexCAN is receiving a message (IDLE=0)

FLT_CONF — Fault Confinement State

This 2-bit field indicates the Confinement State of the FlexCAN module, as shown in [Table 350](#). If the LOM bit in the Control Register is asserted, the FLT_CONF field will indicate “Error Passive”. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted.

Table 350. Fault Confinement State

Value	Meaning
00	Error Active
01	Error Passive
1X	Bus Off

BOFF_INT — ‘Bus Off’ Interrupt

This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect.

1 = FlexCAN module entered ‘Bus Off’ state
 0 = No such occurrence

ERR_INT — Error Interrupt

This bit indicates that at least one of the Error Bits (bits 16-21) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to '1'. Writing '0' has no effect.

- 1 = Indicates setting of any Error Bit in the Error and Status Register
- 0 = No such occurrence

WAK_INT — Wake-Up Interrupt

When FlexCAN is in Stop Mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR Register is set, an interrupt is generated to the CPU. This bit is cleared by writing it to '1'. Writing '0' has no effect.

- 1 = Indicates a recessive to dominant transition received on the CAN bus when the FlexCAN module is in Stop Mode
- 0 = No such occurrence

26.3.4.9 Interrupt Masks 1 Register (IMASK1)

This register allows to enable or disable any number of a range of 32 Message Buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (i.e., when the corresponding IFLAG1 bit is set).

		Base + 0x0028															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF
W		31M	30M	29M	28M	27M	26M	25M	24M	23M	22M	21M	20M	19M	18M	17M	16M
RESET:		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF
W		15M	14M	13M	12M	11M	10M	9M	8M	7M	6M	5M	4M	3M	2M	1M	0M
RESET:		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 325. Interrupt Masks 1 Register (IMASK1)

BUF31M–BUF0M — Buffer MB_i Mask

Each bit enables or disables the respective FlexCAN Message Buffer (MB0 to MB31) Interrupt.

- 1 = The corresponding buffer Interrupt is enabled
- 0 = The corresponding buffer Interrupt is disabled

NOTE

Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set.

26.3.4.10 Interrupt Flags 1 Register (IFLAG1)

This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing it to ‘1’. Writing ‘0’ has no effect.

When the FEN bit in the MCR is set (FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used.

		Base + 0x0030															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF
W		31I	30I	29I	28I	27I	26I	25I	24I	23I	22I	21I	20I	19I	18I	17I	16I
RESET:		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF	BUF
W		15I	14I	13I	12I	11I	10I	9I	8I	7I	6I	5I	4I	3I	2I	1I	0I
RESET:		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 326. Interrupt Flags 1 Register (IFLAG1)

BUF31I–BUF8I — Buffer MB_i Interrupt

Each bit flags the respective FlexCAN Message Buffer (MB8 to MB31) interrupt.

- 1 = The corresponding MB has successfully completed transmission or reception
- 0 = No such occurrence

BUF7I — Buffer MB7 Interrupt or “FIFO Overflow”

If the FIFO is not enabled, this bit flags the interrupt for MB7. If the FIFO is enabled, this flag indicates an overflow condition in the FIFO (frame lost because FIFO is full).

- 1 = MB7 completed transmission/reception or FIFO overflow
- 0 = No such occurrence

BUF6I — Buffer MB6 Interrupt or “FIFO Warning”

If the FIFO is not enabled, this bit flags the interrupt for MB6. If the FIFO is enabled, this flag indicates that 5 out of 6 buffers of the FIFO are already occupied (FIFO almost full).

- 1 = MB6 completed transmission/reception or FIFO almost full
- 0 = No such occurrence

BUF5I — Buffer MB5 Interrupt or “Frames available in FIFO”

If the FIFO is not enabled, this bit flags the interrupt for MB5. If the FIFO is enabled, this flag indicates that at least one frame is available to be read from the FIFO.

- 1 = MB5 completed transmission/reception or frames available in the FIFO
- 0 = No such occurrence

BUF4I–BUF0I — Buffer MB_i Interrupt or “reserved”

If the FIFO is not enabled, these bits flag the interrupts for MB0 to MB4. If the FIFO is enabled, these flags are not used and must be considered as reserved locations.

- 1 = Corresponding MB completed transmission/reception
- 0 = No such occurrence

26.3.4.11 Rx Individual Mask Registers (RXIMR0–RXIMR31)

These registers are used as acceptance masks for ID filtering in Rx MBs and the FIFO. If the FIFO is not enabled, one mask register is provided for each available Message Buffer, providing ID masking capability on a per Message Buffer basis. When the FIFO is enabled (FEN bit in MCR is set), the first 8 Mask Registers apply to the 8 elements of the FIFO filter table (on a one-to-one correspondence), while the rest of the registers apply to the regular MBs, starting from MB8.

The Individual Rx Mask Registers are implemented in RAM, so they are not affected by reset and must be explicitly initialized prior to any reception. Furthermore, they can only be accessed by the CPU while the module is in Freeze Mode. Out of Freeze Mode, write accesses are blocked and read accesses will return “all zeros”. Furthermore, if the BCC bit in the MCR Register is negated, any read or write operation to these registers results in access error.

NOTE

The individual Rx Mask per Message Buffer feature may not be available in low cost MCUs. Please consult the specific MCU documentation to find out if this feature is supported. If not supported, the RXGMASK, RX14MASK and RX15MASK registers are available, regardless of the value of the BCC bit.

Base + 0x0880–0x097F															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0

Figure 327. Rx Individual Mask Registers (RXIMR0 - RXIMR31)

MI31–MI0 — Mask Bits

For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR).

- 1 = The corresponding bit in the filter is checked against the one received
- 0 = the corresponding bit in the filter is “don’t care”

26.4 Functional Description

26.4.1 Overview

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of up to 32 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see [Section 26.3.2, “Message Buffer Structure”](#)). The memory corresponding to the first 8 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 8 extended IDs or 16 standard IDs or 32 8-bit ID slices), each one with its own individual mask register. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be “active” at a given time if it can participate in the matching and arbitration algorithms that are happening at that time. An Rx MB with a ‘0000’ code is inactive (refer to [Table 347](#)). Similarly, a Tx MB with a ‘1000’ or ‘1001’ code is also inactive (refer to [Table 348](#)). An MB not programmed with ‘0000’, ‘1000’ or ‘1001’ will be temporarily deactivated (will not participate in the current arbitration or matching run) when the CPU writes to the C/S field of that MB (see [Section 26.4.6.1, “Message Buffer Deactivation”](#)).

26.4.2 Transmit Process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

- If the MB is active (transmission pending), write ‘1000’ to the Code field to inactivate the MB but then the pending frame may be transmitted without notification (see [Section 1.5.6.2, “Message Buffer Deactivation”](#)).
- Write the ID word.
- Write the data bytes.
- Write the Length, Control and Code fields of the Control and Status word to activate the MB.

Once the MB is activated in the fourth step, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the Code field in the Control and Status word is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new Code field after transmission depends on the code that was used to activate the MB in step four (see [Table 347](#) and [Table 348](#) in [Section 26.3.2, “Message Buffer Structure”](#)).

26.4.3 Arbitration process

The arbitration process is an algorithm executed by the MBM that scans the whole MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID¹ or the lowest MB number or the highest priority, depending on the LBUF and LPRIO_EN bits on the Control Register. The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame
- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in Idle or Bus Off state and the CPU writes to the C/S word of any MB
- Upon leaving Freeze Mode

When LBUF is asserted, the LPRIO_EN bit has no effect and the lowest number buffer is transmitted first. When LBUF and LPRIO_EN are both negated, the MB with the lowest ID is transmitted first but. If LBUF is negated and LPRIO_EN is asserted, the PRIO bits augment the ID used during the arbitration process. With this extended ID concept, arbitration is done based on the full 32-bit ID and the PRIO bits define which MB should be transmitted first, therefore MBs with PRIO = 000 have higher priority. If two or more MBs have the same priority, the regular ID will determine the priority of transmission. If two or more MBs have the same priority (3 extra bits) and the same regular ID, the lowest MB will be transmitted first.

Once the highest priority MB is selected, it is transferred to a temporary storage space called Serial Message Buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called “move-out” and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in HALT or BUS OFF
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) value is bigger.

26.4.4 Receive Process

To be able to receive CAN frames into the mailbox MBs, the CPU must prepare one or more Message Buffers for reception by executing the following steps:

- If the MB has a pending transmission, write ‘1000’ to the Code field to inactivate the MB, but then the pending frame may be transmitted without notification (see Section 1.5.6.2, “Message Buffer Deactivation”). If the MB already programmed as a receiver, just write ‘0000’ to the Code field of the Control and Status word to keep the MB inactive {statement}.
- Write the ID word

1. Actually, if LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

- Write '0100' to the Code field of the Control and Status word to activate the MB

Once the MB is activated in the third step, it will be able to receive frames that match the programmed ID. At the end of a successful reception, the MB is updated by the MBM as follows:

- The value of the Free Running Timer is written into the Time Stamp field
- The received ID, Data (8 bytes at most) and Length fields are stored
- The Code field in the Control and Status word is updated (see [Table 347](#) and [Table 348](#) in [Section 26.3.2, "Message Buffer Structure"](#))
- A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit

Upon receiving the MB interrupt, the CPU should service the received frame using the following procedure:

- Read the Control and Status word (mandatory – activates an internal lock for this buffer)
- Read the ID field (optional – needed only if a mask was used)
- Read the Data field
- Read the Free Running Timer (optional – releases the internal lock)

Upon reading the Control and Status word, if the BUSY bit is set in the Code field, then the CPU should defer the access to the MB until this bit is negated. Reading the Free Running Timer is not mandatory. If not executed the MB remains locked, unless the CPU reads the C/S word of another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is the one on the Control and Status word to assure data coherency (see [Section 26.4.6, "Data Coherence"](#)).

The CPU should synchronize to frame reception by the status flag bit for the specific MB in one of the IFLAG Registers and not by the Code field of that MB. Polling the Code field does not work because once a frame was received and the CPU services the MB (by reading the C/S word followed by unlocking the MB), the Code field will not return to EMPTY. It will remain FULL, as explained in [Table 347](#). If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary: *never do polling by reading directly the C/S word of the MBs. Instead, read the IFLAG registers.*

Note that the received ID field is always stored in the matching MB, thus the contents of the ID field in an MB may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists an Rx matching MB, provided the SRX_DIS bit in the MCR is not asserted. If SRX_DIS is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the FIFO, the CPU must enable and configure the FIFO during Freeze Mode (see [Section 26.4.7, "Rx FIFO"](#)). Upon receiving the frames available interrupt from FIFO, the CPU should service the received frame using the following procedure:

- Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
- Read the ID field (optional – needed only if a mask was used)

- Read the Data field
- Clear the frames available interrupt (mandatory – release the buffer and allow the CPU to read the next FIFO entry)

26.4.5 Matching Process

The matching process is an algorithm executed by the MBM that scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the 8-entry ID table from FIFO is scanned first and then, if a match is not found within the FIFO table, the other MBs are scanned. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

When the frame is received, it is temporarily stored in a hidden auxiliary MB called Serial Message Buffer (SMB). The matching process takes place during the CRC field of the received frame. If a matching ID is found in the FIFO table or in one of the regular MBs, the contents of the SMB will be transferred to the FIFO or to the matched MB during the 6th bit of the End-Of-Frame field of the CAN protocol. This operation is called “move-in”. If any protocol error (CRC, ACK, etc.) is detected, than the move-in operation does not happen.

For the regular mailbox MBs, an MB is said to be “free to receive” a new frame if the following conditions are satisfied:

- The MB is not locked (see [Section 26.4.6.2, “Message Buffer Lock Mechanism”](#))
- The Code field is either EMPTY or else it is FULL or OVERRUN but the CPU has already serviced the MB (read the C/S word and then unlocked the MB)

If the first MB with a matching ID is not “free to receive” the new frame, then the matching algorithm keeps looking for another free MB until it finds one. If it can not find one that is free, then it will overwrite the last matching MB (unless it is locked) and set the Code field to OVERRUN (refer to [Table 347](#) and [Table 348](#)). If the last matching MB is locked, then the new message remains in the SMB, waiting for the MB to be unlocked (see [Section 26.4.6.2, “Message Buffer Lock Mechanism”](#)).

Suppose, for example, that the FIFO is disabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not “free to receive”, so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are “free to receive”, so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the Code field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

The matching algorithm described above can be changed to be the same one used in previous versions of the FlexCAN module. When the BCC bit in MCR is negated, the matching algorithm stops at the first MB

with a matching ID that it finds, whether this MB is free or not. As a result, the message queuing feature does not work if the BCC bit is negated.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Please refer to [Section 26.3.4.11, “Rx Individual Mask Registers \(RXIMR0–RXIMR31\)”](#). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is “don’t care”. Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed if the BCC bit is asserted and while the module is in Freeze Mode.

FlexCAN also supports an alternate masking scheme with only three mask registers (RGXMASK, RX14MASK and RX15MASK) for backwards compatibility. This alternate masking scheme is enabled when the BCC bit in the MCR Register is negated.

NOTE

The individual Rx Mask per Message Buffer feature may not be available in low cost MCUs. Please consult the specific MCU documentation to find out if this feature is supported. If not supported, the RXGMASK, RX14MASK and RX15MASK registers are available, regardless of the value of the BCC bit.

26.4.6 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in [Section 26.4.2, “Transmit Process”](#) and [Section 26.4.4, “Receive Process”](#). Any form of CPU accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

26.4.6.1 Message Buffer Deactivation

Deactivation is mechanism provided to maintain data coherence when the CPU writes to the Control and Status word of active MBs out of Freeze Mode. Any CPU write access to the Control and Status word of an MB causes that MB to be excluded from the transmit or receive processes during the current matching or arbitration round. The deactivation is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore deactivation of that MB is done.

Even with the coherence mechanism described above, writing to the Control and Status word of active MBs when not in Freeze Mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it can not find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after

FlexCAN has scanned the second. The received frame will be lost even if the second matching MB was “free to receive”.

- If a Tx MB containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN will look for another winner within the MBs that it has not scanned yet. Therefore, it may transmit an MB with ID that may not be the lowest at the time because a lower ID might be present in one of the MBs that it had already scanned before the deactivation.
- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the Code field is not updated.

26.4.6.2 Message Buffer Lock Mechanism

Besides MB deactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an “active not empty” Rx MB, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

NOTE

The locking mechanism only applies to Rx MBs which have a code different than INACTIVE (‘0000’) or EMPTY¹ (‘0100’). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no “free to receive” MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the Code field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the Code field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

NOTE

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior will be honored when the BCC bit is negated.

Deactivation takes precedence over locking. If the CPU deactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB.

26.4.7 Rx FIFO

The receive-only FIFO is enabled by asserting the FEN bit in the MCR. The reset value of this bit is zero to maintain software backwards compatibility with previous versions of the module that did not have the FIFO feature. When the FIFO is enabled, the memory region normally occupied by the first 8 MBs (0x80-0xFF) is now reserved for use of the FIFO engine (see [Section 26.3.3, “Rx FIFO Structure”](#)). Management of read and write pointers is done internally by the FIFO engine. The CPU can read the received frames sequentially, in the order they were received, by repeatedly accessing a Message Buffer structure at the beginning of the memory.

The FIFO can store up to 6 frames pending service by the CPU. An interrupt is sent to the CPU when new frames are available in the FIFO. Upon receiving the interrupt, the CPU must read the frame (accessing an MB in the 0x80 address) and then clear the interrupt. The act of clearing the interrupt triggers the FIFO engine to replace the MB in 0x80 with the next frame in the queue, and then issue another interrupt to the CPU. If the FIFO is full and more frames continue to be received, an OVERFLOW interrupt is issued to the CPU and subsequent frames are not accepted until the CPU creates space in the FIFO by reading one or more frames. A warning interrupt is also generated when 5 frames are accumulated in the FIFO.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of 8 32-bit registers that can be configured to one of the following formats (see also [Section 26.3.3, “Rx FIFO Structure”](#)):

- Format A: 8 extended or standard IDs (including IDE and RTR)
- Format B: 16 standard IDs or 16 extended 14-bit ID slices (including IDE and RTR)
- Format C: 32 standard or extended 8-bit ID slices

NOTE

A chosen format is applied to all 8 registers of the filter table. It is not possible to mix formats within the table.

The eight elements of the filter table are individually affected by the first eight Individual Mask Registers (RXIMR0 - RXIMR7), allowing very powerful filtering criteria to be defined. The rest of the RXIMR, starting from RXIM8, continue to affect the regular MBs, starting from MB8. If the BCC bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO filter table is affected by the legacy mask registers as follows: element 6 is affected by RX14MASK, element 7 is affected by RX15MASK and the other elements (0 to 5) are affected by RXGMASK.

26.4.8 CAN Protocol Related Features

26.4.8.1 Remote Frames

Remote frame is a special kind of frame. The user can program a MB to be a Request Remote Frame by writing the MB as Transmit with the RTR bit set to '1'. After the Remote Request frame is transmitted successfully, the MB becomes a Receive Message Buffer, with the same ID as before.

When a Remote Request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the Code field '1010'. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the RTR bit set, then FlexCAN will transmit a Remote Frame as a response.

A received Remote Request Frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a Remote Request Frame was received and matched an MB, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx MB, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

If the Rx FIFO is enabled (bit FEN set in MCR), FlexCAN will not generate an automatic response for Remote Request Frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).

26.4.8.2 Overload Frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

26.4.8.3 Time Stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

Note that the Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in [Section 26.3.4.2, "Control Register \(CTRL\)"](#).

26.4.8.4 Protocol Timing

Figure 328 shows the structure of the clock generation circuitry that feeds the CAN Protocol Interface (CPI) sub-module. The clock source bit (CLK_SRC) in the CTRL Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock (generally from a PLL). In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

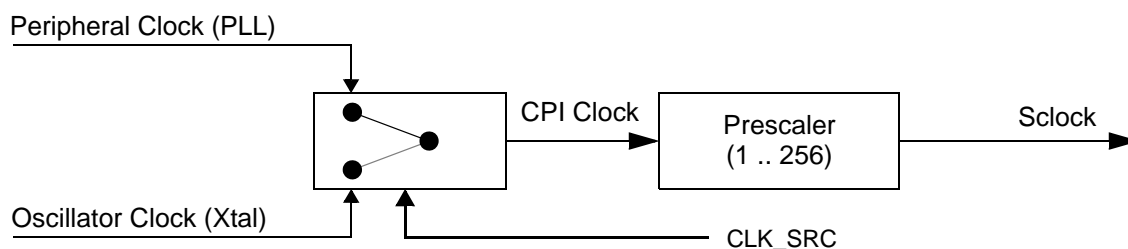


Figure 328. CAN Engine Clocking Scheme

The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

NOTE

This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL clock feeding the FlexCAN module. In these cases, the CLK_SRC bit in the CTRL Register has no effect on the module operation.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRES DIV, PROPSEG, PSEG1, PSEG2 and RJW. See [Section 26.3.4.2, “Control Register \(CTRL\)”](#).

The PRES DIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the ‘time quantum’ used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments¹ (reference [Figure 329](#) and [Table 351](#)):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section

1. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

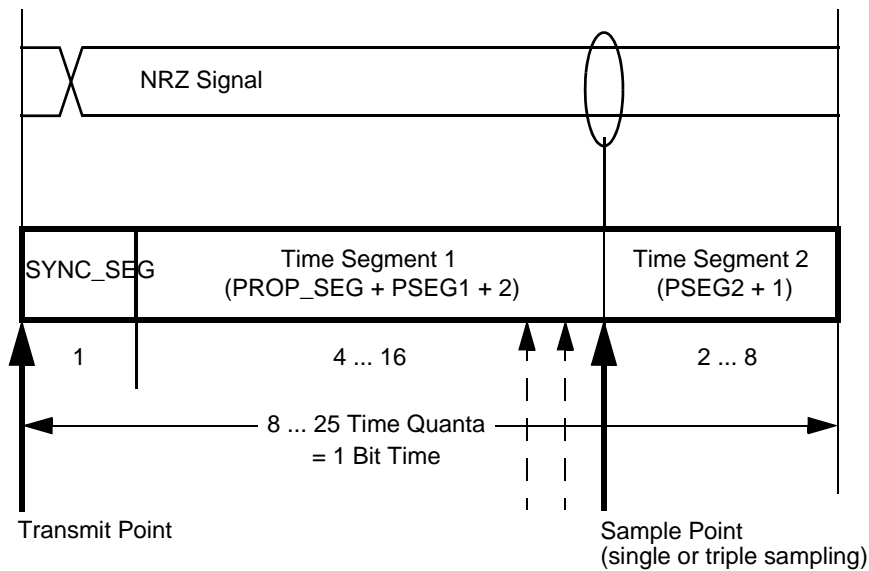


Figure 329. Segments within the Bit Time

Table 351. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 352 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 352. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4

Table 352. CAN Standard Compliant Bit Time Segment Settings

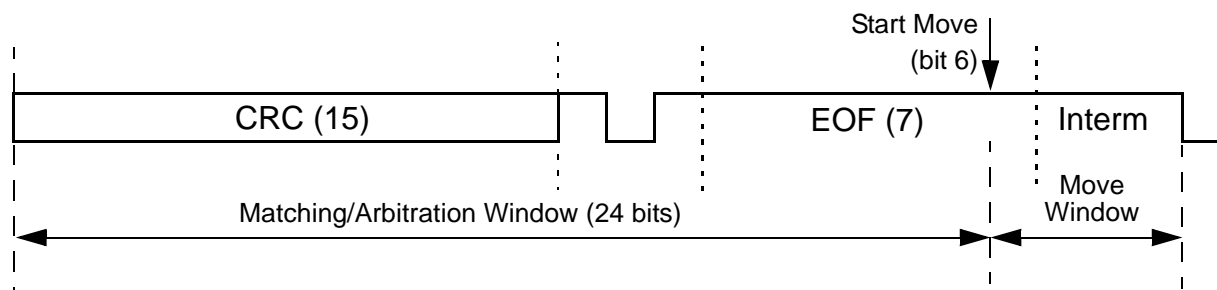
Time Segment 1	Time Segment 2	Re-synchronization Jump Width
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

26.4.8.5 Arbitration and Matching Timing

During normal transmission or reception of frames, the arbitration, matching, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in [Figure 330](#).


Figure 330. Arbitration, Match and Move Time Windows

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in [Table 352](#)
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, i.e. the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in [Table 353](#)

Table 353. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	Minimum Ratio
16	8
32	8
64	16

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in [Table 353](#) can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRES DIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRES DIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

26.4.9 Modes of Operation Details

26.4.9.1 Freeze Mode

This mode is entered by asserting the HALT bit in the MCR Register or when the MCU is put into Debug Mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in any of the low power modes (Disable and Stop). When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible.

Exiting Freeze Mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register
- The MCU is removed from Debug Mode and/or the HALT bit is negated

Once out of Freeze Mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

26.4.9.2 Module Disable Mode

This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive

- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the CPI and MBM sub-modules
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Free Running Timer, the Error Counter Register and the Message Buffers, which cannot be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPM_ACK bit.

26.4.9.3 Stop Mode

This is a system low power mode in which all MCU clocks are stopped for maximum power savings. If FlexCAN receives the global Stop Mode request during Freeze Mode, it sets the LPM_ACK bit, negates the FRZ_ACK bit and then sends a Stop Acknowledge signal to the CPU, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Exiting Stop Mode is done in one of the following ways:

- CPU resuming the clocks and removing the Stop Mode request
- CPU resuming the clocks and Stop Mode request as a result of the Self Wake mechanism

In the Self Wake mechanism, if the SLF_WAK bit in MCR Register was set at the time FlexCAN entered Stop Mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR Register and, if enabled by the WAK_MSK bit in MCR, generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop Mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. [Table 354](#) details the effect of SLF_WAK and WAK_MSK upon wake-up from Stop Mode. Note that wake-up from Stop Mode only works when both bits are asserted.

Table 354. Wake-up from Stop Mode

SLF_WAK	WAK_MSK	MCU Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	No	No
1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Stop Mode. See the WAK_SRC bit in [Section 26.3.4.1, “Module Configuration Register \(MCR\)”](#). This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

NOTE

Not all MCUs are equipped with the low-pass filter. Consult the specific MCU documentation to determine if the low-pass filter is available and to determine its electrical parameters.

26.4.10 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to Ored interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning and Wake Up). The number of actual sources depends on the configured number of Message Buffers.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the CPU writes it to ‘1’ (unless another interrupt is generated at the same time).

NOTE

It must be guaranteed that the CPU only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (bit FEN on MCR set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFLAG1 becomes the “FIFO Overflow” flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the “Frames Available in FIFO flag” and bits 4-0 are unused. See [Section 26.3.4.10, “Interrupt Flags 1 Register \(IFLAG1\)”](#), for more information.

A combined interrupt for all MBs is also generated by an Or of all the interrupt sources from MBs. This interrupt gets generated when any of the MBs generates an interrupt. In this case the CPU must read the IFLAG Registers to determine which MB caused the interrupt.

The other 5 interrupt sources (Bus Off, Error, Tx Warning, Rx Warning and Wake Up) generate interrupts like the MB ones, and can be read from the Error and Status Register. The Bus Off, Error, Tx Warning and Rx Warning interrupt mask bits are located in the Control Register, and the Wake-Up interrupt mask bit is located in the MCR.

26.4.11 Bus Interface

The CPU access to FlexCAN registers are subject to the following rules:

- Read and write access to supervisor registers in User Mode results in access error.

- Read and write access to unimplemented or reserved address space also results in access error. Any access to unimplemented MB or Rx Individual Mask Register locations results in access error. Any access to the Rx Individual Mask Register space when the BCC bit in MCR is negated results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that the Rx Individual Mask Registers can only be accessed in Freeze Mode, and this is still true for unused space within this memory. Note also that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 64 MBs and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The MB memory starts at 0x0060, but the space from 0x0060 to 0x007F is reserved (for SMB usage), and the space from 0x0080 to 0x008F is used by the one MB. This leaves us with the available space from 0x0090 to 0x047F. The available memory in the Mask Registers space would be from 0x0884 to 0x097F.

NOTE

Unused MB space must not be used as general purpose RAM while FlexCAN is transmitting and receiving CAN frames.

26.5 Initialization/Application Information

This section provide instructions for initializing the FlexCAN module.

26.5.1 FlexCAN Initialization Sequence

The FlexCAN module may be reset in three ways:

- MCU level hard reset, which resets all memory mapped registers asynchronously
- MCU level soft reset, which resets some of the memory mapped registers synchronously (refer to [Table 345](#) to see what registers are affected by soft reset)
- SOFT_RST bit in MCR, which has the same effect as the MCU level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) should be selected while the module is in Disable Mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze Mode. In Freeze Mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze Mode (see [Section 26.4.9.1, “Freeze Mode”](#)). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
 - Enable the individual filtering per MB and reception queue features by setting the BCC bit
 - Enable the warning interrupts by setting the WRN_EN bit
 - If required, disable frame self reception by setting the SRX_DIS bit
 - Enable the FIFO by setting the FEN bit
 - Enable the local priority feature by setting the LPRIO_EN bit
- Initialize the Control Register
 - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
 - Determine the bit rate by programming the PRESDIV field
 - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
 - The Control and Status word of all Message Buffers must be initialized
 - If FIFO was enabled, the 8-entry ID table must be initialized
 - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts) and in MCR Register for Wake-Up interrupt
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

26.5.2 FlexCAN Addressing and RAM size configurations

There are RAM configurations that can be implemented within the FlexCAN module.

- For 32 MBs: 544 bytes for MB memory and 128 bytes for Individual Mask Registers

The user can program the maximum number of MBs that will take part in the matching and arbitration processes using the MAXMB field in the MCR Register. For 32 MB configuration, MAXMB can be any number between 0–31.

THIS PAGE INTENTIONALLY BLANK

Chapter 27

Analog-to-Digital Converter (ADC)

27.1 Overview

The Successive Approximation Register Analog-to-Digital Converter (SARADC) digital interface block controls the on-chip SARADC analog block and holds control and status registers accessible for application. It provides accurate and fast conversion data for wide range of applications.

27.2 Introduction

The analog-to-digital converter (ADC) block provides accurate and fast conversions for a wide range of applications. An ADC analog part has its corresponding digital interface (ADCDig).

The ADC digital interface contains advanced features for normal or injected conversion. An injected conversion can be triggered by software or hardware (eTimer). A normal conversion can be triggered by software.

There are two types of input channels:

- Internal ANS (internally multiplexed standard accuracy channels)
- External ANX (externally multiplexed standard accuracy channels)

The mask registers present within the ADCDig can be programmed to configure which channel has to be converted.

A conversion timing register for configuring different sampling and conversion times is associated to each channel type.

Analog watchdogs allow continuous hardware monitoring.

27.2.1 Features

The key features of ADC supported by MPC5604E are:

- 1 ADC unit with 10-bit resolution
- 7 input channels
 - Supports four (channel 0–3) external channels
 - One internally connected channel (channel 4) for the the Temperature sensor
 - One internally connected channel (channel 5) for the internal 3.3 V rail
 - One internally connected channel (channel 6) for the internal 1.2 V rail
- Conversion time < 1 μ s including sampling time at full precision (conversion time target of 700 ns for the analog section)

- 4 analog watchdogs with interrupt capability for continuous hardware monitoring of as many as 4 analog input channels
- Typical sampling time is 150 ns min. (at full precision)
- Sampling and conversion time register CTR0 (internal precision channels, internal standard channels, external channels)
- Right-aligned result format
- One Shot/Scan Modes
- Chain Injection Mode
- Power-down mode
- ADC state machine managing 3 request flows:
 - Regular command
 - Hardware injected command through eTimer
 - Software injected command
- Auto-clock-off
- DMA compatible interface

27.2.2 Block Diagram

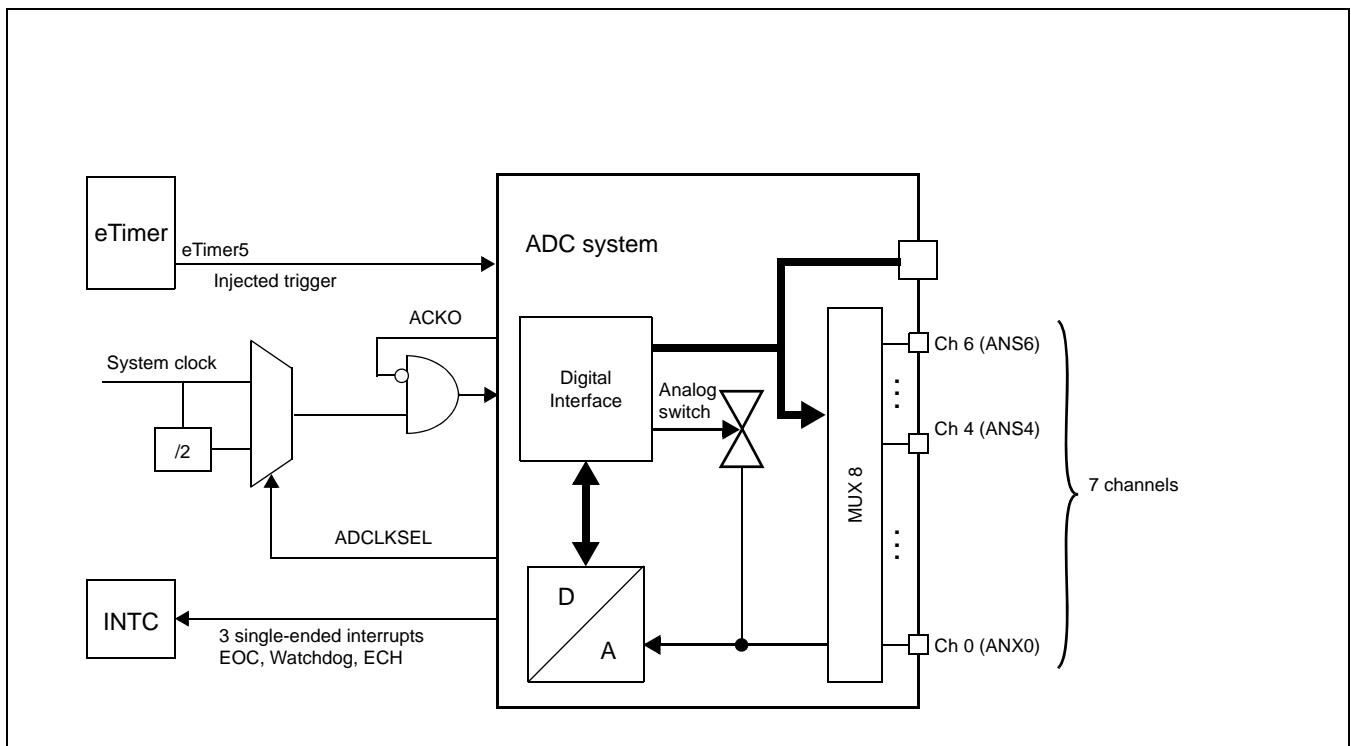


Figure 331. ADC Block Diagram

27.3 Register descriptions

27.3.1 Introduction

Table 355 lists the ADC registers with their address offsets and reset values.

Table 355. ADC digital registers

Address offset (hex.)	Register name	Location
000	Main Configuration Register (MCR)	on page 662
004	Main Status Register (MSR)	on page 665
008 .. 00C	Reserved	—
010	Interrupt Status Register (ISR)	on page 666
014.. 01C	Reserved	—
020	Interrupt Mask Register (IMR)	on page 667
024.. 02C	Reserved	—
030	Watchdog Threshold Interrupt Status Register (WTISR)	on page 669
034	Watchdog Threshold Interrupt Mask Register (WTIMR)	on page 670
038 .. 03C	Reserved	—
040	DMA Enable Register (DMAE)	on page 671
044	DMA Channel Select Register 0 (DMAR0)	on page 672
048..05C	Reserved	—
060	Threshold Register 0 (THRHLR0)	on page 672
064	Threshold Register 1 (THRHLR1)	on page 672
068	Threshold Register 2 (THRHLR2)	on page 672
06C	Threshold Register 3 (THRHLR3)	on page 672
070 .. 090	Reserved	—
094	Conversion Timing Register 0 (CTR0)	on page 673
098	Conversion Timing Register 1 (CTR1)	on page 673
09C .. 0A0	Reserved	—
0A4	Normal Conversion Mask Register 0 (NCMR0)	on page 676
0A8	Reserved	—
0B0		on page 677
0B4	Injected Conversion Mask Register 0 (JCMR0)	on page 677
0BC..0C7	Reserved	—
0C8	Power Down Exit Delay Register (PDEDR)	on page 677
0CC..17C	Reserved	—

Table 355. ADC digital registers (continued)

Address offset (hex.)	Register name	Location
100	Channel 0 Data Register (CDR0)	on page 678
104	Channel 1 Data Register (CDR1)	on page 678
108	Channel 2 Data Register (CDR2)	on page 678
10C	Channel 3 Data Register (CDR3)	on page 678
110	Channel 4 Data Register (CDR4)	on page 678
114	Channel 5 Data Register (CDR5)	on page 678
118	Channel 6 Data Register (CDR6)	on page 678
11C - 2AF	Reserved	—
2B0	Channel Watchdog Selection Register 0 (CWSELR0)	on page 679
2B4 .. 2DC	Reserved	—
2E0	Channel Watchdog Enable Register 0 (CWENR0)	on page 680

Table 356. Bit access descriptions

Access type	Description
read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to these bits.
write 1 to clear (w1c)	Software can clear bits by writing '1'.

27.3.2 Control logic registers

27.3.2.1 Main Configuration Register (MCR)

The Main Configuration Register (MCR) provides configuration settings for the ADC.

Address Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWREN	WLSIDE	MODE	EDGLEV	TRGEN	EDGE	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 332. Main Configuration Register (MCR)
Table 357. Main Configuration Register (MCR) field descriptions

Bit	Description																				
0	OWREN: Overwrite enable This bit enables or disables the functionality to overwrite unread converted data. 0 Prevents overwrite of unread converted data; new result is discarded. 1 Enables converted data to be overwritten by a new conversion.																				
1	WLSIDE: Write left/right-aligned 0 The conversion data is written right-aligned. 1 Data is left-aligned (from 15 to (15 – resolution + 1)).																				
2	MODE: One Shot/Scan 0 One Shot Mode—Configures the normal conversion of one chain. 1 Scan Mode—Configures continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.																				
3	EDGLEV: Edge or level selection for external start trigger 0 Edge configuration for external trigger usage. 1 Level configuration for external trigger usage.																				
4	TRGEN: External trigger enable. This bit must be set to use external triggering to start a conversion. 0 An external trigger cannot be used to start a conversion. 1 An external trigger can start a conversion.																				
5	Start trigger edge/ level detection. The following table shows the interaction between the EDGE bit and the TRGEN and EDGLEV bits. <table border="1" style="margin-left: auto; margin-right: auto;"> <caption>Table 0-1</caption> <thead> <tr> <th>TRGEN</th> <th>EDGLEV</th> <th>EDGE</th> <th>Trigger Detection</th> </tr> </thead> <tbody> <tr> <td>0</td> <td><i>n</i></td> <td><i>n</i></td> <td>External triggering disabled</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>External trigger on falling edge of trigger</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>External trigger on rising edge of trigger</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>External trigger on low edge of trigger</td> </tr> </tbody> </table>	TRGEN	EDGLEV	EDGE	Trigger Detection	0	<i>n</i>	<i>n</i>	External triggering disabled	1	0	0	External trigger on falling edge of trigger	1	0	1	External trigger on rising edge of trigger	1	1	0	External trigger on low edge of trigger
TRGEN	EDGLEV	EDGE	Trigger Detection																		
0	<i>n</i>	<i>n</i>	External triggering disabled																		
1	0	0	External trigger on falling edge of trigger																		
1	0	1	External trigger on rising edge of trigger																		
1	1	0	External trigger on low edge of trigger																		

Table 357. Main Configuration Register (MCR) field descriptions (continued)

Bit	Description
6	Reserved Must be kept at 0.
7	<p>NSTART: Normal Start conversion Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation. This bit stays high while the conversion is ongoing (or pending during injection mode). 0 Causes the current chain conversion to finish and stops the operation. 1 Starts the chain or scan conversion.</p> <p>Note: While scan chain conversion is ongoing, and NSTART bit is cleared to stop the chain, it is possible that chain conversion ends after one more chain conversion after the completion of chain during which NSTART bit was cleared.</p>
8	Reserved Write of any value has no effect, read value is always 0.
9	<p>JTRGEN: Injection external trigger enable 0 External trigger disabled for channel injection (injected conversion cannot be started using an external signal). 1 External trigger enabled for channel injection.</p>
10	<p>JEDGE: Injection trigger edge selection Edge selection for external trigger, if JTRGEN = 1. 0 Selects falling edge for the external trigger. 1 Selects rising edge for the external trigger.</p>
11	<p>JSTART: Injection start Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted.</p>
12:13	Reserved Write of any value has no effect, read value is always 0.
14	Reserved Must be kept at 0.
15:22	Reserved Write of any value has no effect, read value is always 0.
23	<p>ADCLKSEL: Analog clock frequency selector If this bit is set the AD_clk frequency is equal to ipg_clk frequency. Otherwise, it is half of ipg_clk frequency. This bit can be written in power-down only.</p>
26	<p>ACKO: Auto-clock-off enable If set, this bit enables the Auto clock off feature. 0 Auto clock off is disabled. 1 Auto clock off is enabled.</p>
27:28	Reserved Must be kept at 0.

Table 357. Main Configuration Register (MCR) field descriptions (continued)

Bit	Description
29:30	Reserved Write of any value has no effect, read value is always 0.
31	PWDN: Power-down enable When this bit is set, the analog module is requested to enter Power Down mode. When ADC status is PWDN, resetting this bit starts ADC transition to IDLE mode. 0 ADC is in normal mode. 1 ADC has been requested to power down.

NOTE

After resetting PWDN bit, wait for the power up time suggested by ADC power up parameter in the DS (t_{ADC_PU}) before any conversion is initiated.

27.3.2.2 Main Status Register (MSR)

The Main Status Register (MSR) provides status bits for the ADC.

Address Base + 0x0004

Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	N START	0	0	0	J START	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR[0:6]				0	0	0	ACK 0	0	0	0	ADCSTATUS[0:2]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 333. Main Status Register (MSR)
Table 358. Main Status Register (MSR) field descriptions

Field	Description
0:6	Reserved Write of any value has no effect, read value is always 0.
7	NSTART This status bit is used to signal that a Normal conversion is ongoing.
8:10	Reserved Write of any value has no effect, read value is always 0.

Table 358. Main Status Register (MSR) field descriptions (continued)

Field	Description
11	JSTART This status bit is used to signal that an Injected conversion is ongoing.
12:14	Reserved Write of any value has no effect, read value is always 0.
15	Reserved Write of any value has no effect, read value is always 0.
16:22	CHADDR[0:6]: Channel under measure address This status bit is used to signal which channel is under measure.
23:25	Reserved Write of any value has no effect, read value is always 0.
26	ACKO: Auto-clock-off enable This status bit is used to signal if the Auto-clock-off feature is on.
27:28	Reserved Write of any value has no effect, read value is always 0.
29:31	ADCSTATUS[0:2] The value of this parameter depends on ADC status: 000 IDLE 001 Power-down 010 Wait state 011 — 100 Sample 101 — 110 Conversion 111 —

27.3.3 Interrupt registers

27.3.3.1 Interrupt Status Register (ISR)

The Interrupt Status Register (ISR) contains interrupt status bits for the ADC.

Address Base + 0x0010

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	JEO C	JEC H	EOC	ECH
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 334. Interrupt Status Register (ISR)
Table 359. Interrupt Status Register (ISR) field descriptions

Field	Description
0:24	Reserved Write of any value has no effect, read value is always 0.
25:26	Reserved Write of any value has no effect, read value is always 0.
27	Reserved Write of any value has no effect, read value is always 0
28	End of Injected Channel Conversion interrupt (JEOC) flag. It is the interrupt of the digital end of conversion for the injected channel; active when set. When this bit is set, a JEOC interrupt has occurred.
29	End of Injected Chain Conversion interrupt (JECH) flag. It is the interrupt of the digital end of chain conversion for the injected channel; active when set. When this bit is set, a JECH interrupt has occurred.
30	End of Channel Conversion interrupt (EOC) flag. It is the interrupt of the digital end of conversion. When this bit is set, an EOC interrupt has occurred.
31	End of Chain Conversion interrupt (ECH) flag. It is the interrupt of the digital end of chain conversion. When this bit is set, an ECH interrupt has occurred.

27.3.3.2 Interrupt Mask Register (IMR)

The Interrupt Mask Register (IMR) contains the interrupt enable bits for the ADC.

Address Base + 0x0020

Access: User read/write

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

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

Figure 335. Interrupt Mask Register (IMR)

Table 360. Interrupt Mask Register (IMR) field descriptions

Field	Description
0:24	Reserved Write of any value has no effect, read value is always 0.
25:26	Reserved Must be kept at 0.
27	Reserved Must be kept at 0.
28	MSKJEOC: Mask bit for JEOC When set, the JEOC interrupt is enabled.
29	MSKJECH: Mask bit for JECH When set, the JECH interrupt is enabled.
30	MSKEOC: Mask bit for EOC When set, the EOC interrupt is enabled.
31	MSKECH: Mask bit for ECH When set, the ECH interrupt is enabled.

27.3.3.3 Watchdog Threshold Interrupt Status Register (WTISR)

Address Base + 0x0030

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	WDG 3H	WDG 3L	WDG 2H	WDG 2L	WDG 1H	WDG 1L	WDG 0H	WDG 0L
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 336. Watchdog Threshold Interrupt Status Register (WTISR)
Table 361. Watchdog Threshold Interrupt Status Register (WTISR) field descriptions

Field	Description
0:23	Reserved Write of any value has no effect, read value is always 0.
24	WDG3H This corresponds to the status flag generated on the converted value being higher than the programmed higher threshold.
25	WDG3L This corresponds to the status flag generated on the converted value being lower than the programmed lower threshold.
26	WDG2H This corresponds to the status flag generated on the converted value being higher than the programmed higher threshold.
27	WDG2L This corresponds to the status flag generated on the converted value being lower than the programmed lower threshold.
28	WDG1H This corresponds to the status flag generated on the converted value being higher than the programmed higher threshold.
29	WDG1L This corresponds to the status flag generated on the converted value being lower than the programmed lower threshold.

Table 361. Watchdog Threshold Interrupt Status Register (WTISR) field descriptions (continued)

Field	Description
30	WDG0H This corresponds to the status flag generated on the converted value being higher than the programmed higher threshold.
31	WDG0L This corresponds to the status flag generated on the converted value being lower than the programmed lower threshold.

27.3.3.4 Watchdog Threshold Interrupt Mask Register (WTIMR)

Reset value: 0x0000_0000

Address Base + 0x0034

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	MSK WDG 3H	MSK WDG 3L	MSK WDG 2H	MSK WDG 2L	MSK WDG 1H	MSK WDG 1L	MSK WDG 0H	MSK WDG 0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 337. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 362. Watchdog Threshold Interrupt Mask Register (WTIMR) field descriptions

Field	Description
0:23	Reserved Write of any value has no effect, read value is always 0.
24	MSKWDG3H This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. When set the interrupt is enabled.
25	MSKWDG3L This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. When set the interrupt is enabled.
26	MSKWDG2H This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. When set the interrupt is enabled.

Table 362. Watchdog Threshold Interrupt Mask Register (WTIMR) field descriptions (continued)

Field	Description
27	MSKWDOG2L This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. When set the interrupt is enabled.
28	MSKWDOG1H This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. When set the interrupt is enabled.
29	MSKWDOG1L This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. When set the interrupt is enabled.
30	MSKWDOG0H This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. When set the interrupt is enabled.
31	MSKWDOG0L This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. When set the interrupt is enabled.

27.3.4 DMA registers

27.3.4.1 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the ADC.

Address Base + 0x0040

Access: User read/write

:																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCL	DMA
W															R	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 338. DMA Enable Register (DMAE)

Table 363. DMA Enable Register (DMAE) field descriptions

Field	Description
0:29	Reserved Write of any value has no effect, read value is always 0.
30	DCLR: DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers
31	DMAEN: DMA global enable 0 DMA feature is disabled. 1 DMA feature is enabled.

27.3.4.2 DMA Channel Select Register 0 (DMAR0)

Reset value: 0x0000_0000

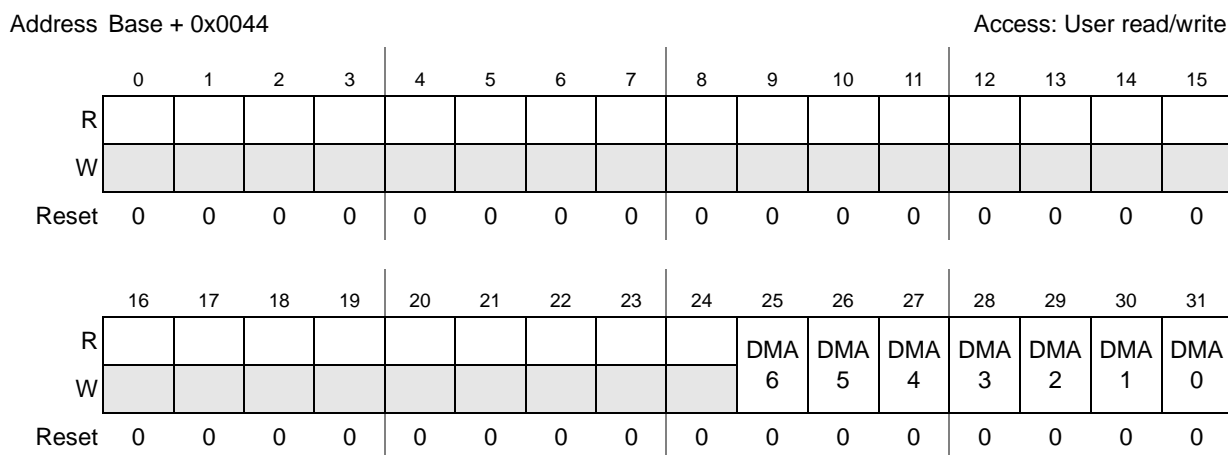


Figure 339. DMA Channel Select Register (DMAR)

Table 364. DMA Channel Select Register 0 (DMAR0) field descriptions

Field	Description
DMA _n	DMA _n : DMA enable When set (DMA _n = 1), channel n is enabled to transfer data in DMA mode.

27.3.5 Threshold registers

27.3.5.1 Introduction

These four registers are used to store the user programmable lower and upper thresholds' values. The inverter bit and the mask bit for mask the interrupt are stored in the TRC registers.

27.3.5.2 Threshold Register (THRHLR[0:3])

The four THRHLR_n registers are used to store the user-programmable thresholds' 10-bit values.

Address Base + 0x0060 (THRHLR0)
 Base + 0x0064 (THRHLR1)
 Base + 0x0068 (THRHLR2)
 Base + 0x006C (THRHLR3)

Access: User read/write

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

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

Figure 340. Threshold Register (THRHLR[0:3])

Table 365. Threshold Register (THRHLR[0:3]) field descriptions

Field	Description
0:5	Reserved Write of any value has no effect, read value is always 0.
6:15	THRH: High threshold value for channel <i>n</i> .
16:21	Reserved Write of any value has no effect, read value is always 0.
22:31	THRL: Low threshold value for channel <i>n</i> .

27.3.6 Conversion timing registers CTR[0..1]

CTR0 = associated to external channels (from 0 to 3)

CTR1 = associated to internal channels (from 4 to 6)

Address Base + 0x0094 (CTR0)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INPLATCH	0	OFFSHIFT [0:1]		0	INPCMP [0:1]		0	INPSAMP[0:7]							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

Figure 341. Conversion timing registers CTR[0..1] (CTR0)

Table 366. Conversion timing registers CTR[0..1] (CTR0) field descriptions

Field	Description
0:15	Reserved Write of any value has no effect, read value is always 0.
16	INPLATCH Configuration bit for latching phase duration
17	Reserved Write of any value has no effect, read value is always 0.
18:19	OFFSHIFT[0:1] Configuration for offset shift characteristic 00 No shift (that is the transition between codes 000h and 001h) is reached when the A_{VIN} (analog input voltage) is equal to 1 LSB. 01 Transition between code 000h and 001h is reached when the A_{VIN} is equal to 1/2 LSB 10 Transition between code 00h and 001h is reached when the A_{VIN} is equal to 0 11 Not used
20	Reserved Write of any value has no effect, read value is always 0.
21:22	INPCMP[0:1] Configuration bits for comparison phase duration
23	Reserved Write of any value has no effect, read value is always 0.
24:31	INPSAMP[0:7] Configuration bits for sampling phase duration

Address Base + 0x0098 (CTR1)

Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0			0			0								
W	INP LATCH		OFFSHIFT [0:1]			INPCMP [0:1]			INPSAMP[0:6]						TSENSOR_SEL	
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

Figure 342. Conversion timing registers CTR[0..1] (CTR1)
Table 367. Conversion timing registers CTR[0..1] (CTR1) field descriptions

Field	Description
0:15	Reserved Write of any value has no effect, read value is always 0.
16	INPLATCH Configuration bit for latching phase duration
17	Reserved Write of any value has no effect, read value is always 0.
18:19	OFFSHIFT[0:1] Configuration for offset shift characteristic 00 No shift (that is the transition between codes 000h and 001h) is reached when the A_{VIN} (analog input voltage) is equal to 1 LSB. 01 Transition between code 000h and 001h is reached when the A_{VIN} is equal to 1/2 LSB 10 Transition between code 00h and 001h is reached when the A_{VIN} is equal to 0 11 Not used
20	Reserved Write of any value has no effect, read value is always 0.
21:22	INPCMP[0:1] Configuration bits for comparison phase duration
23	Reserved Write of any value has no effect, read value is always 0.

Table 367. Conversion timing registers CTR[0..1] (CTR1) field descriptions (continued)

Field	Description
24:30	INPSAMP[0:6] Configuration bits for sampling phase duration
31	TSENSOR_SEL 0) PTAT mode 1) CTAT mode

27.3.7 Mask registers

27.3.7.1 Introduction

These registers are used to program which of the 8 input channels must be converted during Normal and Injected conversion.

27.3.7.2 Normal Conversion Mask Register 0 (NCMR0)

Reset value: 0x0000_0000

Address Base + 0x00A4

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 343. Normal Conversion Mask Register 0 (NCMR0)

Table 368. Normal Conversion Mask Register 0 (NCMR0) field descriptions

Field	Description
CHn	CHn: Sampling enable. When set (CHn = 1), sampling is enabled for channel n.

27.3.7.3 Injected Conversion Mask Register 0 (JCMR0)

Reset value: 0x0000_0000

Address Base + 0x00B4												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0							
W										CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 344. Injected Conversion Mask Register 0 (JCMR0)

Table 369. Injected Conversion Mask Register 0 (JCMR0) field descriptions

Field	Description
CHn	CHn: Sampling enable. When set (CHn = 1), sampling is enabled for channel n.

27.3.8 Power Down Exit Delay Register (PDEDR)

Reset value: 0x0000_0000

Address Base + 0x00B4												Access: User read/write				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PDED							0	0	0	0	0	0	0	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 345. Power Down Exit Delay Register (PDEDR)

Table 370. Power Down Exit Delay Register (PDEDR) field descriptions

Field	Description
0:7	PDED The delay between the power down bit reset and the starting of conversion. The power down delay is calculated as (PDED X 1/Frequency of ADC HardMacro clock)
8:31	Reserved Write of any value has no effect, read value is always zero.

27.3.9 Data registers

27.3.9.1 Introduction

ADC conversion results are stored in data registers. There is one register per channel.

CDR[0...3] = External channels

CDR[4..6] = Internal channels

Each data register also gives information regarding the corresponding result as described below.

27.3.9.2 Channel Data Register (CDR[0..6])

Address See [Table 355](#)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	VA LID	OVE RW	RESULT [0:1]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CDATA[0:9]									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 346. Channel Data Register (CDR[0..6])
Table 371. CDR[0..6] field descriptions

Field	Description
0:11	Reserved Write of any value has no effect, read value is always 0.
12	VALID Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table 371. CDR[0..6] field descriptions (continued)

Field	Description
13	OVERW: Overwrite data This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, then OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, then OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14:15	RESULT[0:1] This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode. 01 Data is a result of Injected conversion mode. 10 Reserved. 11 Reserved.
16:21	Reserved Write of any value has no effect, read value is always 0
22:31	CDATA[0:9]: Channel converted data

27.3.9.3 Channel Watchdog Select Register (CWSELR0)

This register selects the threshold register which provides the values to be used for upper and lower bounds for channel n.

Address: Base + 0x2B0

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WSEL_CH0				WSEL_CH1				WSEL_CH2				WSEL_CH3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WSEL_CH4				WSEL_CH5				WSEL_CH6				0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 347. Channel Watchdog Select Register 0 (CWSELR0)
Table 372. CWSELR0 field descriptions

Field	Description
WSEL_CHn	Channel Watchdog select for channel n

27.3.9.4 Channel Watchdog Enable Register (CWENR0)

Address: Base + 0x2E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CWEN0	CWEN1	CWEN2	CWEN3	CWEN4	CWEN5	CWEN6	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 348. Channel Watchdog Enable Register (CWENR0)

Table 373. CWENR0 field descriptions

Field	Description
CWENn	CWENn: Channel Watchdog enable When set (CWENn = 1) Watchdog feature is enabled for channel n.

27.4 Functional description

27.4.1 Analog channel conversion

Two conversion modes are available within the ADCDig:

- Normal conversion
- Injected conversion

27.4.1.1 Normal conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (NCMR). Each channel can be individually enabled by setting ‘1’ in the corresponding field of NCMR registers. Mask registers must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (NSTART bit in the Main Status Register (MSR) is reset).

27.4.1.2 Start of normal conversion

The normal conversion can be started in two ways:

- By software —If the external trigger enable bit is reset, the conversion chain starts when the NSTART bit in the MCR is set. Once normal conversion gets started, any further write operation of NSTART bit of MCR will not have any effect during ongoing conversion.

The NSTART status bit in the MSR is automatically set when the normal conversion starts. At the same time the NSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running conversion is completed.

If the content of all the normal conversion mask registers is zero (that is, no channel is selected) the conversion operation is considered completed and the interrupt ECH is immediately issued after the start of conversion.

27.4.1.3 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot
- Scan

To enter one of these modes, it is necessary to program the MODE bit in the MCR. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in [Figure 349](#).

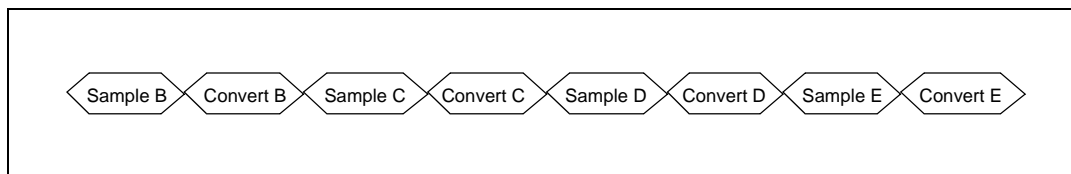


Figure 349. Normal conversion flow

One Shot Mode:

In One Shot Mode (MODE = 0), a sequential conversion specified in the NCMR registers is performed only once. At the end of each conversion, the digital result of the conversion is stored in the corresponding data register.

Example 1. One Shot Mode (MODE = 0)

Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the One Shot Mode. MODE = 0 is set for One Shot mode. Conversion starts from the channel B followed by conversion of channels D-E. At the end of conversion of channel E the scanning of channels stops.

The NSTART status bit in the MSR is automatically set when the Normal conversion starts. At the same time the NSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running conversion is completed.

NOTE

Ensure that a new conversion sequence is not started when the current conversion is ongoing. This can be done by issuing a new conversion setting, MCR[NSTART] only when MSR[NSTART] = 0, where, MSR[NSTART] indicates the present status of conversion.

MSR[NSTART] = 1 means that a conversion is ongoing and
MSR[NSTART] = 0 means that the previous conversion is finished.

Scan Mode:

In Scan Mode (MODE = 1), a sequential conversion of N channels specified in the NCMR registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

The NSTART status bit in the MSR is automatically set when the Normal conversion starts. Unlike One Shot Mode, the NSTART bit in the MCR is not reset. It can be reset by software when the user needs to stop scan mode. In that case, the ADC completes the current scan conversion and, after the last conversion, also resets the NSTART bit in the MSR.

Example 2. Scan Mode (MODE = 1)

Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the Scan Mode. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself till the NSTART bit in the MCR is reset by software.

If the conversion is started by an external trigger and EDGLEV is '0', the NSTART bit in the MCR is not set. As a consequence, once started the only way to stop scan mode conversion is to set the MODE bit to '0'.

At the end of each conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit) and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit).

27.4.1.4 Injected channel conversion

A conversion chain can be injected into the ongoing Normal conversion by configuring the Injected Conversion Mask Registers (JCMR). As Normal conversion, each channel can be individually selected. This injected conversion can only occur in One Shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped as shown in Figure 350.

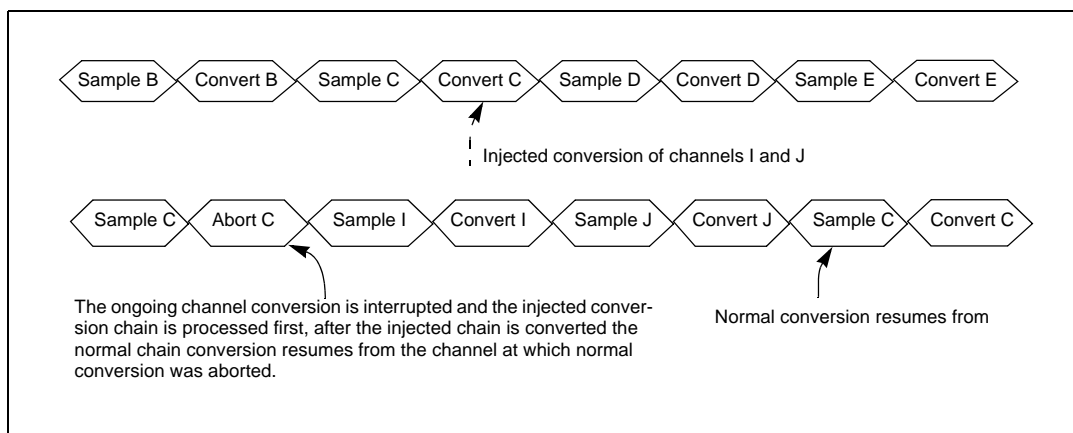


Figure 350. Injected sample/conversion sequence

The Injected conversion can be started in two ways:

- Software — The conversion can be started by setting the JSTART bit in the MCR; the current conversion is suspended and the injected chain is converted. At the end of the chain, the JSTART bit in the MSR is reset and the normal chain conversion is resumed. The JSTART status bit in the MSR is automatically set when the Injected conversion starts. At the same time the JSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running injected conversion is completed.
- Hardware — Injection can be started by giving a rising or falling edge on the channel 5 of eTimer.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

If the content of all the injected conversion mask registers is zero (that is, no channel is selected) the interrupt JECH is immediately issued after the start of conversion.

27.4.2 Analog clock generator and conversion timings

The clock frequency can be selected by programming the ADCLKSEL bit in the MCR. When this bit is set to '1' the ADC clock has the same frequency as the system clock. Otherwise, the ADC clock is half of the system clock frequency. The ADCLKSEL bit can be written only in power-down mode.

When the internal divider is not enabled ($ADCCLKSEL = 1$), it is important that the associated clock divider in the clock generation module is '1'. This is needed to ensure 50% clock duty cycle.

The direct clock should basically be used only in low power mode when the device is using only the 16 MHz fast internal RC oscillator, but the conversion still requires a 16 MHz clock (an 8 MHz clock is not fast enough to achieve the minimum conversion time. Refer to [Section 27.4.3, “ADC sampling and conversion timing”](#)).

In all other cases, the ADC should use the clock divided by two internally.

27.4.3 ADC sampling and conversion timing

In order to support different loading and switching times, several different Conversion Timing registers (CTR) are present. There is one register per channel type. INPLATCH and INPCMP configurations are limited when the system clock frequency is greater than 20 MHz.

When a conversion is started, the ADC connects the internal sampling capacitor to the respective analog input pin, allowing the capacitance to charge up to the input voltage value. The time to load the capacitor is referred to as sampling time. After completion of the sampling phase, the evaluation phase starts and all the bits corresponding to the resolution of the ADC are estimated to provide the conversion result.

The conversion times are programmed via the bit fields of the CTR. Bit fields INPLATCH, INPCMP and INPSAMPLE are used to define the total conversion duration (T_{conv}) and in particular the partition between sampling phase duration (T_{sample}) and total evaluation phase duration (T_{eval}).

The sampling phase duration is:

$$T_{\text{sample}} = (\text{INPSAMPLE} - \text{ndelay}) \cdot T_{\text{ck}}$$

$$\text{INPSAMPLE} \geq 3$$

where ndelay is equal to 0.5 if INPSAMPLE is less than or equal to 06h, otherwise it is 1. INPSAMPLE must be greater than or equal to 3 (hardware requirement).

The total evaluation phase duration is:

$$T_{\text{eval}} = 10 \cdot T_{\text{biteval}} = 10 \cdot (\text{INPCMP} \cdot T_{\text{ck}})$$

$$(\text{INPCMP} \geq 1) \quad \text{and} \quad (\text{INPLATCH} < \text{INPCMP})$$

INPCMP must be greater than or equal to 1 and INPLATCH must be less than INPCMP (hardware requirements).

The total conversion duration is (not including external multiplexing):

$$T_{\text{conv}} = T_{\text{sample}} + T_{\text{eval}} + (\text{ndelay} \cdot T_{\text{ck}})$$

The timings refer to the unit T_{ck} , where $f_{\text{ck}} = (1/2 \times \text{ADC peripheral set clock})$. The maximum clock frequency is specified in [Table 374](#).

Table 374. Max AD_clk frequency and related configuration settings

INPLATCH	INPCMP	INPSAMPLE	AD_clk f _{max} (MHz)	T _{sample} min. (ns)
0	1h	3h	20	125
0	1h	4h	20 + 4%	168
1	2h	4h	20 + 4%	168
1	2h	5h	20 + 4%	135
1	3h	6h	32 + 4%	132
1	3h	7h	40 + 4%	128
1	3h	8h	50 + 4%	134
1	3h	9h	60 + 4%	128

[Table 375](#) lists the possible combinations by configuring the AD_clk at 60 MHz.

Table 375. ADC sampling and conversion timing

INPLATCH	INPCMP	INPSAMP	T _{sample} ¹	T _{eval}	ndelay	T _{conv}
1	11	0000 1001	8 * Tck	30 * Tck	1 * Tck	39 * Tck ²
1	11	0000 1010	9 * Tck	30 * Tck	1 * Tck	40 * Tck
1	11	0000 1011	10 * Tck	30 * Tck	1 * Tck	41 * Tck

Table 375. ADC sampling and conversion timing (continued)

INPLATCH	INPCMP	INPSAMP	T_{sample}^1	T_{eval}	ndelay	T_{conv}
1	11	0000 1100	11 * Tck	30 * Tck	1 * Tck	42 * Tck
1	11	0000 1101	12 * Tck	30 * Tck	1 * Tck	43 * Tck
1	11	0000 1110	13 * Tck	30 * Tck	1 * Tck	44 * Tck
1	11	0000 1111	14 * Tck	30 * Tck	1 * Tck	45 * Tck
...
1	11	1111 1100	251 * Tck	30 * Tck	1 * Tck	282 * Tck
1	11	1111 1101	252 * Tck	30 * Tck	1 * Tck	283 * Tck
1	11	1111 1110	253 * Tck	30 * Tck	1 * Tck	284 * Tck
1	11	1111 1111	254 * Tck	30 * Tck	1 * Tck	285 * Tck

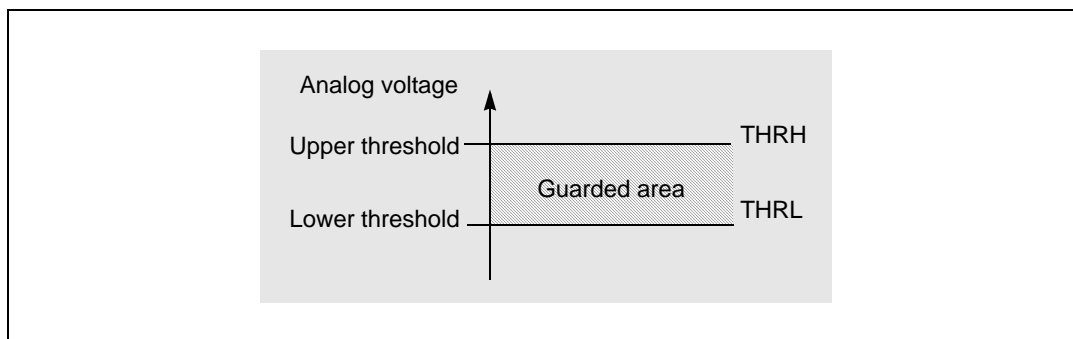
¹ Represents the number of clock cycles that this operation will last

² The ADC minimum conversion time at 60 MHz frequency is 39 * Tck; that corresponds to 650 ns.

27.4.4 Programmable analog watchdog

27.4.4.1 Introduction

The analog watchdogs are used for determining whether the result of a channel conversion lies within a given guarded area (as shown in [Figure 351](#)) specified by an upper and a lower threshold value named THRH and THRL respectively.


Figure 351. Guarded area

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guarded area then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in [Table 376](#). Depending on the mask bits MSKWDOGxL and MSKWDOGxH in the WTIMR, an interrupt is generated on threshold violation.

Table 376. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRL <= converted data <= THRH

The channel on which the analog watchdog is to be applied is selected by the THRCH field in the TRC registers. The analog watchdog is enabled by setting the corresponding THREN bit in the same register.

The lower and higher threshold values for the analog watchdog are programmed using the registers THRHLR.

For example, if channel number 3 is to be monitored with threshold values in THRHLR1, then the THRCH field is programmed in the TRC1 register to select channel number 3.

A set of threshold registers (THRHLRx and TRCx) can be linked only to a single channel for a particular THRCH value. If another channel is to be monitored with same threshold values, then the THRCH field in the TRCx register has to be programmed again.

NOTE

If the higher threshold for the analog watchdog is programmed lower than the lower threshold and the converted value is less than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the higher threshold) then the interrupt WDGxH for high threshold violation is set. Thus, the user should avoid that situation as it could lead to misinterpretation of the watchdog interrupts.

27.4.5 DMA functionality

A DMA request can be programmed after the conversion of every channel by setting the respective masking bit in the DMAR registers. The DMAR masking registers must be programmed before starting any conversion. There is one DMAR per channel type.

The DMA transfers can be enabled using the DMAEN bit of DMAE register. When the DCLR bit of DMAE register is set then the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

27.4.6 Interrupts

The ADC generates the following maskable interrupt signals:

- ADC_EOC interrupt requests
 - EOC (end of conversion)
 - ECH (end of chain)
 - JEOC (end of injected conversion)

- JECH (end of injected chain)
- WDGxL and WDGxH (watchdog threshold) interrupt requests

Interrupts are generated during the conversion process to signal events such as End Of Conversion as described in the [Section 27.3.3.1, “Interrupt Status Register \(ISR\)”](#). Two registers named Interrupts Status Register (ISR) and Interrupt Mask Register (IMR) are provided to check and enable the interrupt request to EIC module.

Interrupts can be individually enabled on a channel by channel base by programming the IMR (Interrupt Mask Register).

The analog watchdog interrupts are handled by two registers 32-bit WTISR (Watchdog Threshold Interrupt Status Register) and WTIMR (Watchdog Threshold Interrupt Mask Register) in order to check and enable the interrupt request to the EIC module. The Watchdog interrupt source sets two pending bits WDGxH and WDGxL in the WTISR for each of the four channels being monitored.

27.4.7 Power-down mode

The analog part of the ADC can be put in low power mode by setting the PWDN bit in the MCR. After releasing the reset signal the ADC analog module is kept in power-down mode by default, so this state must be exited before starting any operation by resetting the appropriate bit in the MCR.

The power-down mode can be requested at any time by setting the PWDN bit in the MCR. If a conversion is ongoing, the ADC hard macrocell cannot immediately enter the power-down mode. In fact, the ADC enters power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit.

Bit ADCSTATUS[0] in the MSR is set only when ADC enters power-down mode.

After the power-down phase is completed the process ongoing before the power-down phase must be restarted manually (by setting the appropriate START bit).

Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

27.4.8 Auto-clock-off mode

To reduce power consumption during the IDLE mode of operation (without going into power-down mode), an “auto-clock-off” feature can be enabled by setting the ACKO bit in the MCR. When enabled, the analog clock is automatically switched off when no operation is ongoing, that is, no conversion is programmed by the user.

THIS PAGE INTENTIONALLY BLANK

Chapter 28

Enhanced Motor Control Timer (eTimer)

28.1 Introduction

28.1.1 Overview

Each eTimer module contains 6 identical counter/timer channels and one watchdog timer function. Each 16 bit counter/timer channel contains a prescaler, a counter, a load register, a hold register, two queued capture registers, two compare registers, two compare preload registers, and four control registers.

NOTE

This document uses the terms “Timer” and “Counter” interchangeably because the counter/timers may perform either or both tasks.

The Load register provides the initialization value to the counter when the counter’s terminal value has been reached. For true modulo counting the counter can also be initialized by the CMPLD1 or CMPLD2 registers.

The Hold register captures the counter’s value when other counters are being read. This feature supports the reading of cascaded counters coherently.

The Capture registers enable an external signal to take a “snap shot” of the counter’s current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The Prescaler provides different time bases useful for clocking the counter/timer.

The Counter provides the ability to count internal or external events.

Within the eTimer module (set of 6 timer/counter channels) the input pins are shareable.

28.1.2 Features

The eTimer module design includes these distinctive features:

- 6 bit counters/timers.
- Count up/down.
- Counters are cascadable.
- Enhanced programmable up/down modulo counting.

- Max count rate equals peripheral clock/2 for external clocks.
- Max count rate equals peripheral clock for internal clocks.
- Count once or repeatedly.
- Counters are preloadable.
- Compare registers are preloadable.
- Counters can share available input pins.
- Separate prescaler for each counter.
- Each counter has capture and compare capability.
- Continuous and single shot capture for enhanced speed measurement.
- DMA support of capture registers and compare registers.
- 32 bit watchdog capability to detect stalled quadrature counting.
- OFLAG comparison for safety critical applications.
- Programmable operation during debug mode and stop mode.
- Programmable input filter.
- Counting start can be synchronized across counters.

NOTE

eTimer counter must be initialized before start counting SAI sync pulses.

28.1.3 Customization

This section specifies where to find chip specific parameters.

Table 377. Customization References for Specific Chips

Feature or Parameter	Where to find
Base address	The base address for the location of the timer's memory mapped registers. It can be found in the memory section of the system specification.
Input filter	The chip spec must specify if input filtering is included for external inputs.
Synchronization	The chip spec must specify if the ENBL register is included.
Watchdog	The chip spec must specify if the watchdog function is included.
Number of channels	The chip spec must specify the number of timer channels from 2 to 8
DMA	The chip spec must specify whether DMA requests/accesses are supported, which types, and how many
Capture FIFO depth	The chip spec must specify the depth of the capture FIFOs.
Auxiliary signals	The chip spec must specify the presence and number of auxiliary signals from 0 up to the number of channels.

28.1.4 Module Block Diagram

The eTimer block diagram is shown in [Figure 352](#).

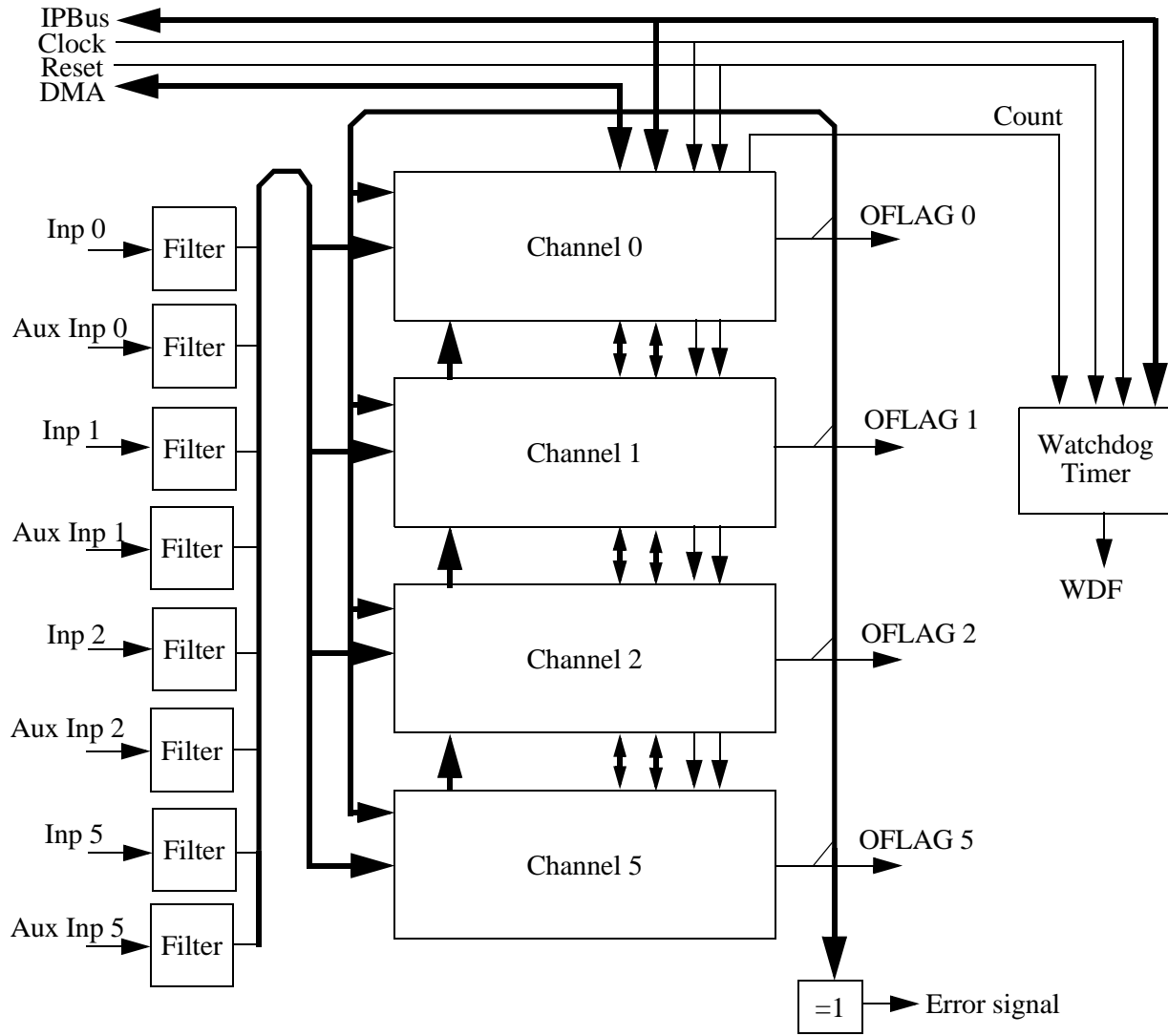


Figure 352. eTimer Block Diagram

28.1.5 Channel Block Diagram

Each of the timer/counter channels within the eTimer are shown in [Figure 353](#).

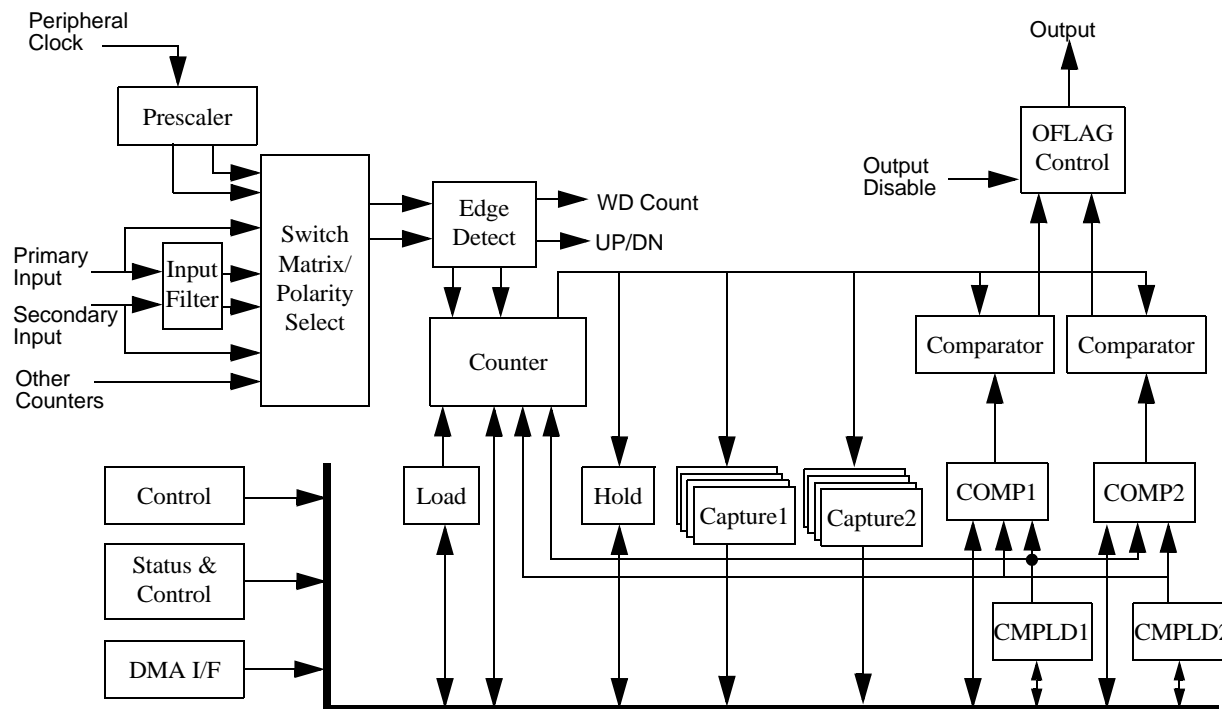


Figure 353. eTimer Channel Block Diagram

28.2 External Signal Descriptions

The eTimer module has up to 6 external signals that can be used as either inputs or outputs. The external interface signals are generated using the `ipp_do_tmr`, `ipp_obe_tmr`, and `ipp_ind_tmr` signals. There is one auxiliary input. The eTimer also interfaces to the Peripheral Bus.

28.2.1 TIO[5:0] - Timer Input/Outputs

These pins can be independently configured to be either timer input sources or output flags.

28.2.2 TAI[2] - Timer Auxiliary Input

DSPI1 SCK signal, which is connected to Auxiliary input pin can act as an alternate input choice for the timer channels.

28.3 Functional Description

28.3.1 General

Each channel has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.

- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a “count with direction” format.
- The counter’s terminal count value (modulo) is programmable.
 - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count “rolls over” to zero.

The external inputs to each counter/timer are shareable among each of the up to eight channels within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be “captured”
- They can be used to generate interrupt requests

The polarity of the external inputs is selectable.

The primary output of each channel is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is programmable.

Any channel can be assigned as a “Master”. A master’s compare signal can be broadcasted to the other channels within the module. The other channels can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a Master channel’s compare event occurs.

28.3.2 Counting Modes

The selected external signals are sampled at the eTimer’s base clock rate and then run through a transition detector. The maximum count rate is one-half of the eTimer’s base clock rate when using an external signal. Internal clock sources can be used to clock the counters at the eTimer’s base clock rate.

If a counter is programmed to count to a specific value and then stop, the CNTMODE field in the CTRL1 register is cleared when the count terminates.

28.3.2.1 STOP Mode

If the CNTMODE field is set to ‘000’, the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

28.3.2.2 COUNT Mode

If the CNTMODE field is set to ‘001’, the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as “widgets” on a conveyor belt passing a sensor. If the selected input is inverted by setting the PIPS bit, then the negative edge of the selected external input signal is counted.

See [Section 28.3.2.9, “CASCADE-COUNT Mode”](#), through [Section 28.3.2.12, “VARIABLE-FREQUENCY PWM Mode”](#), for additional capabilities of this operating mode.

28.3.2.3 EDGE-COUNT Mode

If the CNTMODE field is set to ‘010’, the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment such as a simple encoder wheel.

28.3.2.4 GATED-COUNT Mode

If the CNTMODE field is set to ‘011’, the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the PIPS bit, then the counter will count while the selected secondary input is low.

28.3.2.5 QUADRATURE-COUNT Mode

If the CNTMODE field is set to ‘100’, the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

[Figure 354](#) shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

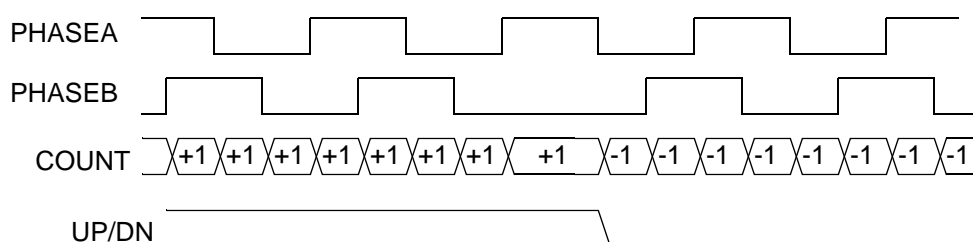


Figure 354. Quadrature Incremental Position Encoder

28.3.2.6 SIGNED-COUNT Mode

If the CNTMODE field is set to ‘101’, the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

28.3.2.7 TRIGGERED-COUNT Mode

If the CNTMODE field is set to ‘110’, the counter will begin counting the primary clock source after a positive transition (negative if SIPS = 1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. Subsequent secondary positive input transitions will continue to restart and stop the counting until a compare event occurs.

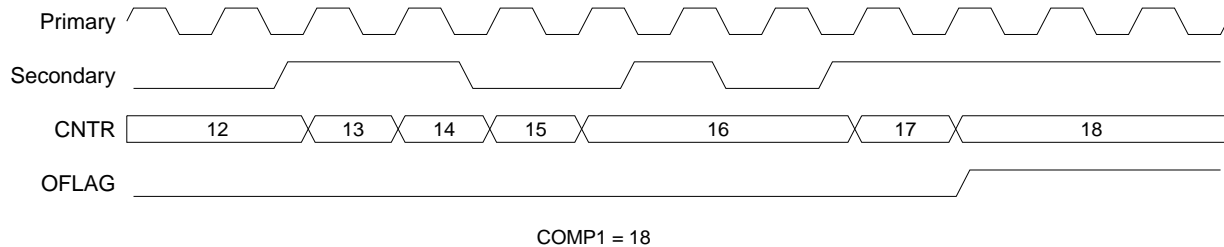


Figure 355. Triggered Count Mode (Length=1)

28.3.2.8 ONE-SHOT Mode

If the CNTMODE field is set to ‘110’, and the counter is set to reinitialize at a compare event (LENGTH =1), and the OFLAG OUTMODE is set to ‘0101’ (cleared on init, set on compare), the counter works in a “One-Shot Mode”. An external events causes the counter to count, when terminal count is reached, the output is asserted. This “delayed” output can be used to provide timing delays.

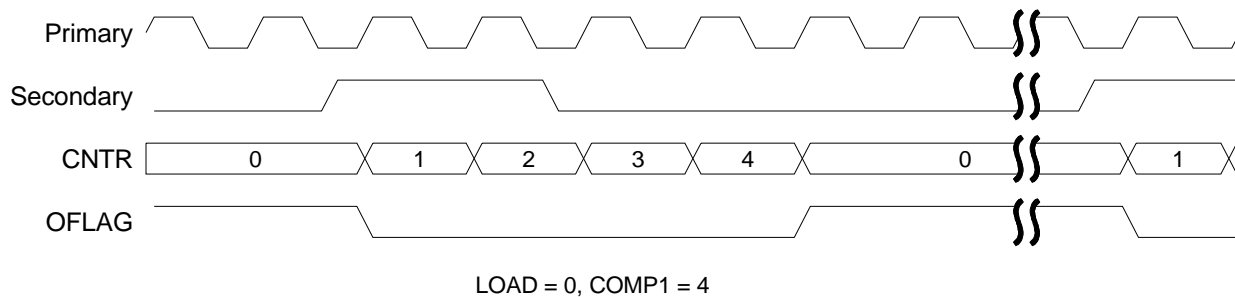


Figure 356. One-Shot Mode (Length=1)

28.3.2.9 CASCADE-COUNT Mode

If the CNTMODE field is set to ‘111’, the counter’s input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This “Cascade” or “Daisy-Chained” mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to two counters may be cascaded to create a 32 bit wide synchronous counter.

Whenever any counter is read within a counter module, all of the counters’ values within the module are captured in their respective HOLD registers. This action supports the reading of a cascaded counter chain.

First read any counter of a cascaded counter chain, then read the HOLD registers of the other counters in the chain. The cascaded counter mode is synchronous.

NOTE

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a “ripple” mode, where higher order counters will transition a clock later than a purely synchronous design.

NOTE

A channel can be cascaded with any other channel, but you can’t cascade more than 2 channels together. You can create separate cascades of pairs of channels. For example, you can cascade channels 0 and 1 and separately cascade channels 6 and 5. You can’t cascade channels 0, 1, and 5.

28.3.2.10 PULSE-OUTPUT Mode

If the counter is setup for CNTMODE = 001, and the OFLAG OUTMODE is set to ‘1111’ (gated clock output), and the ONCE bit is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

NOTE

This does not work if the PRISRC is set to 11000 (IP_bus/1).

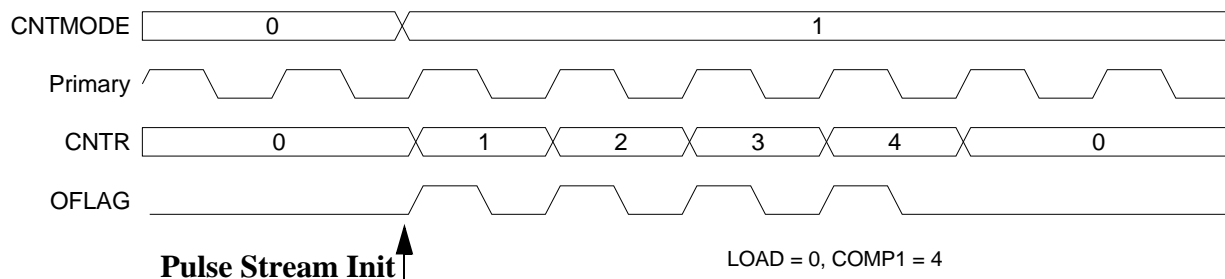


Figure 357. Pulse Output Mode

28.3.2.11 FIXED-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count through roll-over (LENGTH = 0), continuous count (ONCE = 0) and the OFLAG OUTMODE is ‘0111’ (set on compare, cleared on counter roll-over) then the counter output yields a Pulse Width Modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

28.3.2.12 VARIABLE-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count till compare (LENGTH = 1), continuous count (ONCE = 0) and the OFLAG OUTMMODE is ‘0100’ (toggle OFLAG and alternate compare registers) then the

counter output yields a Pulse Width Modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

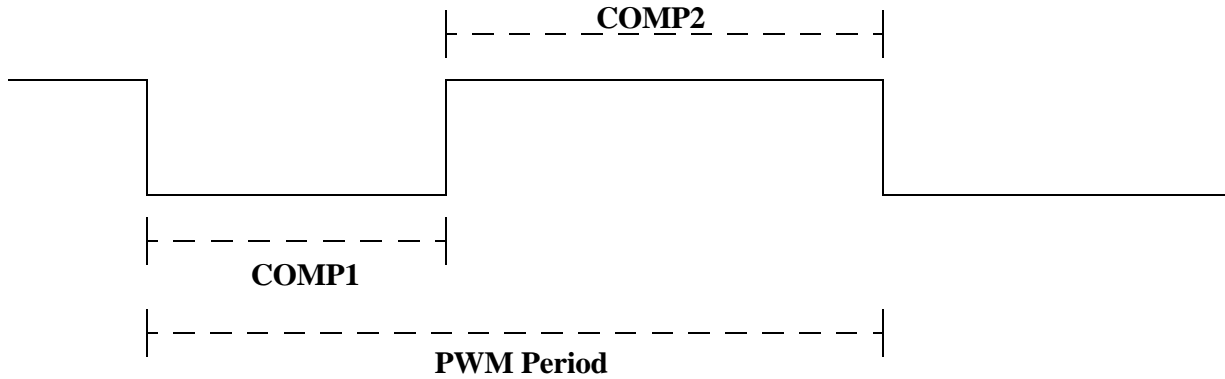
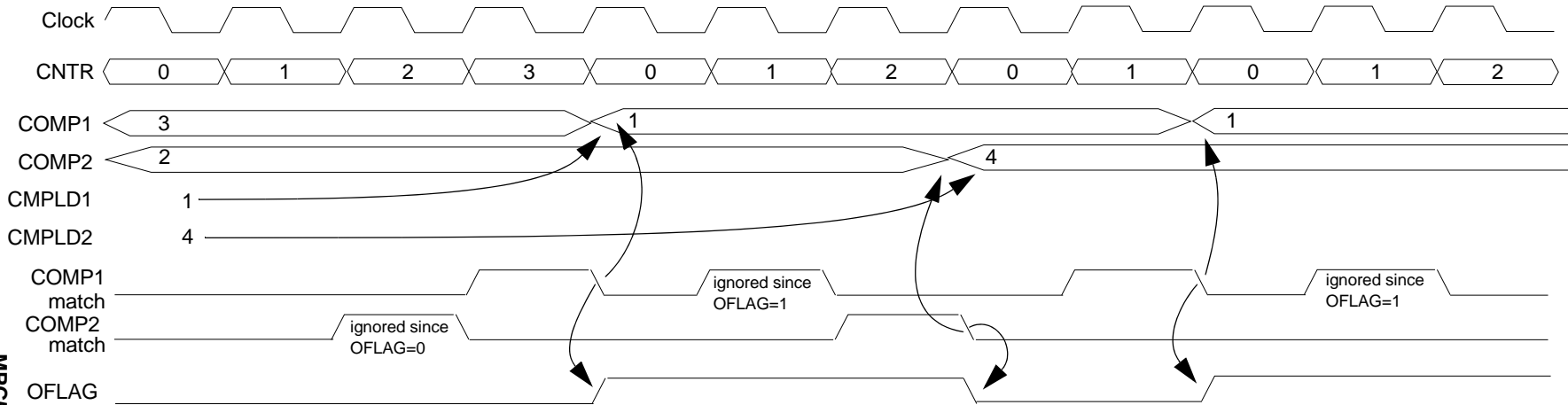


Figure 358. Variable PWM Waveform



CLC1=010, load COMP1 when CNTR=COMP1
 CLC2=101, load COMP2 when CNTR=COMP2

Figure 359. Variable Frequency PWM Mode Timing

28.3.2.13 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability.

The COMP1 register should be set to the desired maximum count value or \$FFFF to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or \$0000 to indicate the minimum unsigned value prior to roll-under.

If the output mode is set to 0100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time. COMP1 is used when OFLAG=0 and COMP2 is used when OFLAG=1. OFLAG can be forced to a value using CTRL2[FORCE] and CTRL2[VAL].

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to \$FFFF or \$0000, roll over, then begin counting toward the new value. The check is: $CNTR = COMP_x$, *not* $CNTR > COMP1$ or $CNTR < COMP2$.

The use of the CMPLD1 and CMPLD2 registers to preload compare values will help to minimize this problem.

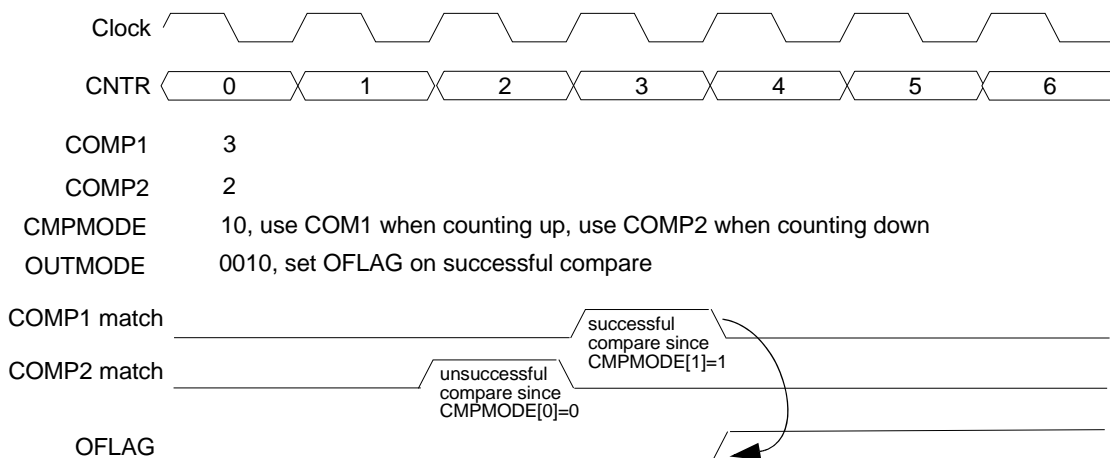


Figure 360. Compare Register and OFLAG Timing

28.3.2.14 Usage of Compare Load Registers

The CMPLD1, CMPLD2 and CCCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. A compare register can be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there is the possibility that the counter may have already counted past the new compare value by the time the compare register is updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are updated in hardware in the same way the counter register is re-initialized to the value stored in the LOAD register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See [Figure 358](#).

28.3.2.15 MODULO COUNTING Mode

To create a modulo counter using COMP1 and COMP2 as the counter boundaries (instead of \$0000 and \$FFFF), set the registers in the following manner. Set CNTMODE to either 100 (quadrature count mode)

or 101 (count with direction mode). Use count through roll-over (LENGTH = 0) and continuous count (ONCE = 0). Set COMP1 and CMPLD1 to the upper boundary value. Set COMP2 and CMPLD2 to the lower boundary value. Set CMPMODE = 10 (COMP1 is used when counting up and COMP2 is used when counting down). Set CLC2 = 110 (load CNTR with value of CMPLD2 on COMP1 compare) and CLC1 = 111 (load CNTR with value of CMPLD1 on COMP2 compare).

28.3.2.16 Compare Register and OFLAG Operation

The compare flags and output flag are registered signals and can only change state after the counter reached the compare value. This means that, if the counter is continuously counting, then the compare flag is set on the count after the counter reaches the compare value. When counting is not continuous (such as when using a prescaler or when counting edges on the primary input), the compare flag or OFLAG are not updated until the counter is about to transition to the next count. This means that if the compare value is 4, then the flags won't be changed until the clock edge that the counter goes from 4 to the next value.

Counting positive edges of primary input, COMP1==4

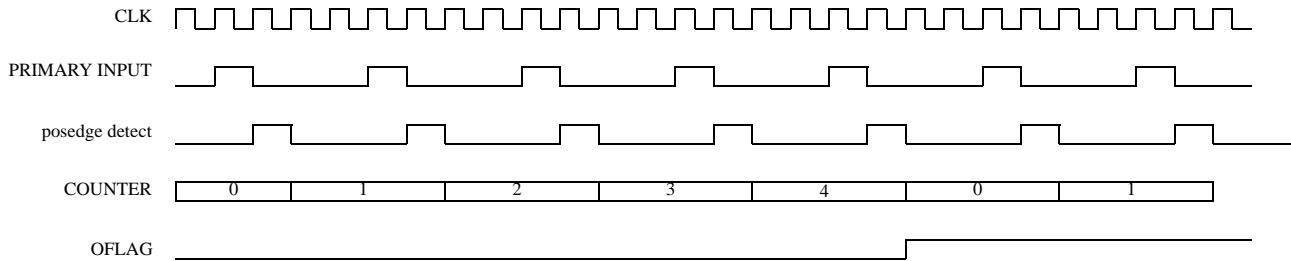


Figure 361. OFLAG timing during non-continuous counting

Counting positive edges of clock, COMP1==4

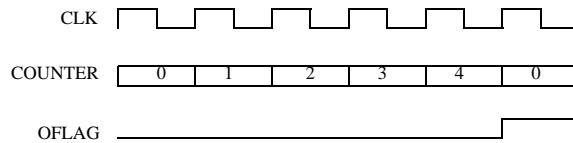


Figure 362. OFLAG timing during continuous counting

28.3.3 Other Features

28.3.3.1 Redundant OFLAG Checking

This mode allows the user to bundle two timer functions generating any pattern to compare their resulting OFLAG behaviors (output signal).

The redundant mode is used to support online checks for functional safety reasons. Whenever a mismatch between the two adjacent channels occurs, it is reported via an interrupt to the core and the two outputs are put into their inactive states. An error is flagged via the RCF flag.

This feature can be tested by forcing a transition on one of the OFLAGs using the VAL and FORCE bits of the channel.

28.3.3.2 Loopback Checking

This mode is always available in that one channel can generate an OFLAG while another channel uses the first channel's OFLAG as its input to be measured and verified to be as expected.

28.3.3.3 Input Capture Mode

Input capture is used to measure pulse width (by capturing the counter value on two successive input edges) or waveform period (by capturing the counter value on two consecutive rising edges or two consecutive falling edges). The Capture Registers store a copy of the counter's value when an input edge (positive, negative, or both) is detected. The type of edge to be captured by each circuit is determined by the CPT1MODE and CPT2MODE bits whose functionality is listed in [Table 390](#). Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

28.3.3.4 Master/Slave Mode

Any timer channel can be assigned as a Master (MSTR = 1). A Master's compare signal can be broadcasted to the other channels within the module. The other counters can be configured to reinitialize their counters (COINIT = 1) and/or force their OFLAG output signals (COFRC = 1) to predetermined values when a Master counter compare event occurs.

28.3.3.5 Watchdog Timer

The watchdog timer is used to monitor for a stalled count when channel 0 is in quadrature count mode. When the watchdog is enabled, it loads the time-out value into a down counter. The down counter counts as long as channel 0 is in quadrature decode count mode. If this down counter reaches zero, an interrupt

is asserted. The down counter is reloaded to the time-out value each time the counter value from channel 0 changes. If the channel 0 count value is toggling between two values (indicating a possibly stalled encoder), then the down counter is not reloaded.

28.4 Memory Map and Registers

28.4.1 Overview

The base address of the eTimer module will differ from chip to chip. All memory mapped registers described below have their location described in relation to the BASE Address.

28.4.2 Module Memory Map

Table 378. eTimer Memory Map

Address	Reg Name	Description
Timer channel registers (repeated for each channel as CHNL goes from 0 to 5)		
eTimer_BASE + (\$20 * CHNL) + \$0	COMP1	Compare Register 1
eTimer_BASE + (\$20 * CHNL) + \$2	COMP2	Compare Register 2
eTimer_BASE + (\$20 * CHNL) + \$4	CAPT1	Capture Register 1
eTimer_BASE + (\$20 * CHNL) + \$6	CAPT2	Capture Register 2
eTimer_BASE + (\$20 * CHNL) + \$8	LOAD	Load Register
eTimer_BASE + (\$20 * CHNL) + \$A	HOLD	Hold Register
eTimer_BASE + (\$20 * CHNL) + \$C	CNTR	Counter Register
eTimer_BASE + (\$20 * CHNL) + \$E	CTRL1	Control Register 1
eTimer_BASE + (\$20 * CHNL) + \$10	CTRL2	Control Register 2
eTimer_BASE + (\$20 * CHNL) + \$12	CTRL3	Control Register 3
eTimer_BASE + (\$20 * CHNL) + \$14	STS	Status Register
eTimer_BASE + (\$20 * CHNL) + \$16	INTDMA	Interrupt and DMA Enable Register
eTimer_BASE + (\$20 * CHNL) + \$18	CMPLD1	Comparator Load Register 1
eTimer_BASE + (\$20 * CHNL) + \$1A	CMPLD2	Comparator Load Register 2
eTimer_BASE + (\$20 * CHNL) + \$1C	CCCTRL	Compare and Capture Control Register
eTimer_BASE + (\$20 * CHNL) + \$1E	FILT	Input Filter Register
Watchdog timer registers		
eTimer_BASE + \$100	WDTOL	Watchdog Time-out Low Register

Table 378. eTimer Memory Map (continued)

Address	Reg Name	Description
eTimer_BASE + \$102	WDTOH	Watchdog Time-out High Register
Configuration Registers		
eTimer_BASE + \$10C	ENBL	Channel Enable Register
eTimer_BASE + \$110	DREQ0	DMA Request 0 Select Register
eTimer_BASE + \$112	DREQ1	DMA Request 1 Select Register

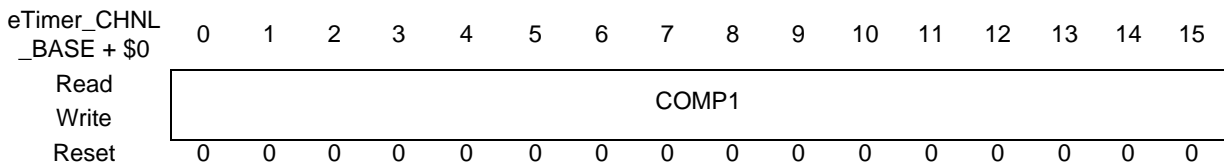
28.4.3 Register Descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level. There are a set of registers for each timer channel, a set for the watchdog timer, and a set of configuration registers.

28.4.4 Timer Channel Registers

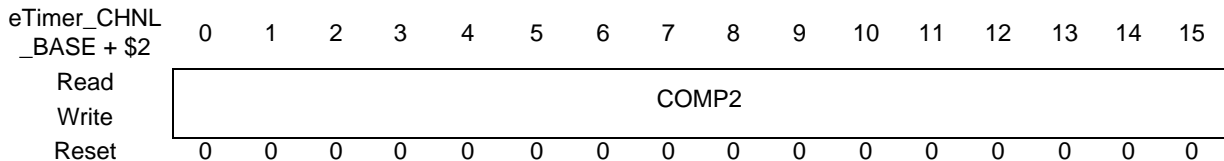
These registers are repeated for each timer channel. The base address of channel 0 is the same as the base address of the eTimer module as a whole. The base address of channel 1 is \$20. This is the base address of the eTimer module plus an offset based on the number of bytes of registers in a timer channel. The base address of each subsequent timer channel is equal to the base address of the previous channel plus this same offset of \$20.

28.4.4.1 Compare Register 1 (COMP1)


Figure 363. Compare Register 1 (COMP1)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP1 can be found in [Section 28.3.2.13, “Usage of Compare Registers”](#).

28.4.4.2 Compare Register 2 (COMP2)


Figure 364. Compare Register 2 (COMP2)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP2 can be found in [Section 28.3.2.13, “Usage of Compare Registers”](#).

28.4.4.3 Capture Register 1 (CAPT1)

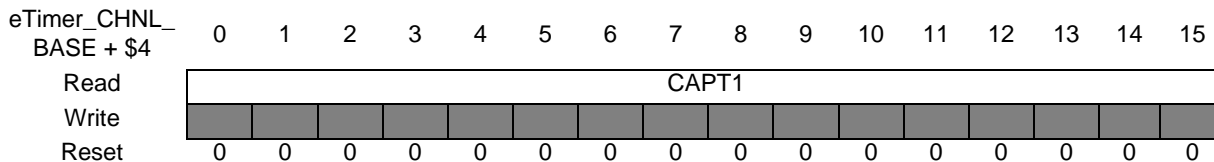


Figure 365. Capture Register 1 (CAPT1)

This read only register stores the value captured from the counter. Exactly when a capture occurs is defined by the CPT1MODE bits. This is actually a 4 deep FIFO and not a single register. This register is not byte accessible.

28.4.4.4 Capture Register 2 (CAPT2)

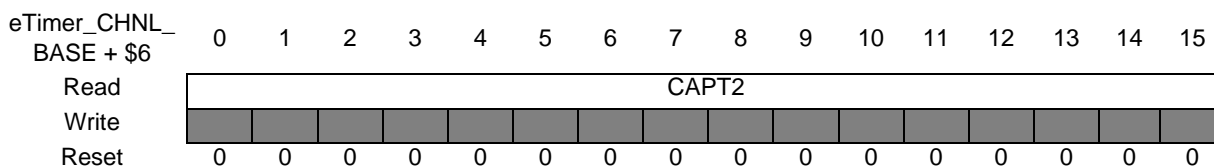


Figure 366. Capture Register 2 (CAPT2)

This read only register stores the value captured from the counter. Exactly when a capture occurs is defined by the CPT2MODE bits. This is actually a four deep FIFO and not a single register. This register is not byte accessible.

28.4.4.5 Load Register (LOAD)

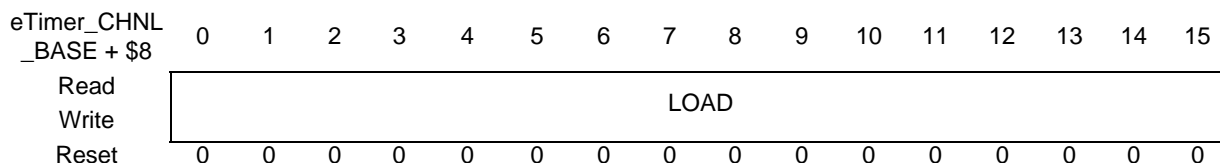


Figure 367. Load Register (LOAD)

This read/write register stores the value used to initialize the counter. This register is not byte accessible.

28.4.4.6 Hold Register (HOLD)

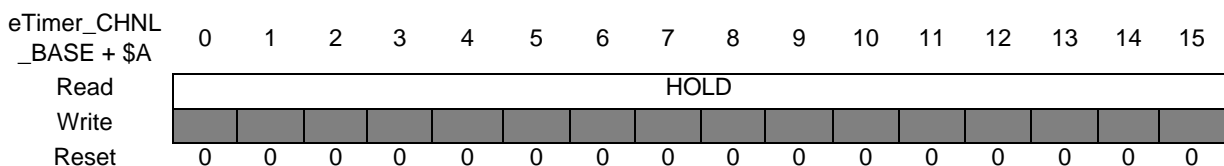


Figure 368. Hold Register (HOLD)

This read only register stores the counter’s value whenever any of the other counters within a module are read. This is used to support coherent reading of cascaded counters.

In addition, this read only register stores the counter’s value if any of the Compare and Capture Control Registers (CCCTRL) within a module are read.

28.4.4.7 Counter Register (CNTR)

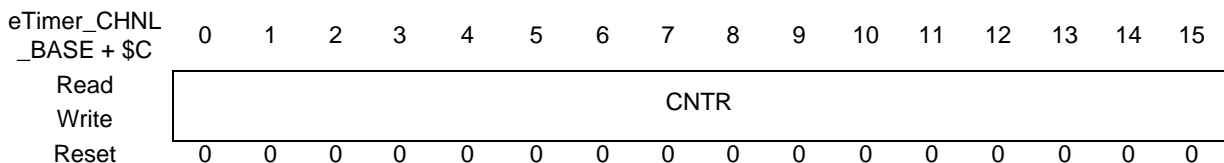


Figure 369. Counter (CNTR)

This read/write register is the counter for this channel of the timer module. This register is not byte accessible.

28.4.4.8 Control Register 1 (CTRL1)

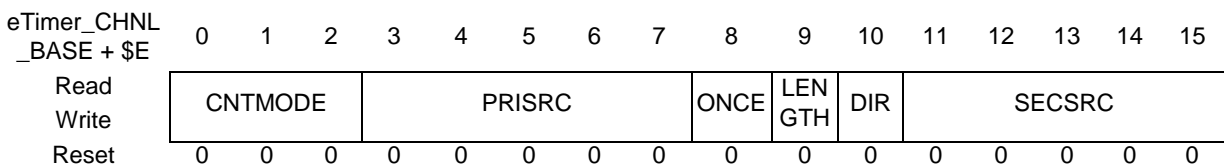


Figure 370. Control Register 1 (CTRL1)

CNTMODE - Count Mode

These bits control the basic counting and behavior of the counter.

Table 379. Count Mode Values

Value	Meaning
000	No Operation
001	Count rising edges of primary source ¹
010	Count rising and falling edges of primary source ²
011	Count rising edges of primary source while secondary input high active

Table 379. Count Mode Values (continued)

Value	Meaning
100	Quadrature count mode, uses primary and secondary sources
101	Count primary source rising edges, secondary source specifies direction (1 = minus) ³
110	Edge of secondary source triggers primary count till compare
111	Cascaded counter mode, up/down ⁴

¹ Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1. If primary count source is IP bus clock, only rising edges are counted regardless of PIPS value.

² IP Bus clock divide by 1 can not be used as a primary count source in edge count mode.

³ Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1.

⁴ Primary count source must be set to one of the counter outputs.

PRISRC - Primary Count Source

These bits select the primary count source.

Table 380. Primary Count Source Values

Value	Meaning
00000	Counter #0 input pin
00001	Counter #1 input pin
00010	Counter #2 input pin
00011	Counter #3 input pin
00100	Counter #4 input pin
00101	Counter #5 input pin
00110	Counter #6 input pin
00111	Counter #7 input pin
01000	Reserved
01001	Reserved
01010	Auxiliary input #2 pin
01011	Reserved
01100	Reserved
01101	Reserved
01110	Reserved
01111	Reserved
10000	Counter #0 output
10001	Counter #1 output
10010	Counter #2 output
10011	Counter #3 output
10100	Counter #4 output
10101	Counter #5 output

Table 380. Primary Count Source Values (continued)

Value	Meaning
10110	Counter #6 output
10111	Counter #7 output
11000	IP Bus clock divide by 1 prescaler
11001	IP Bus clock divide by 2 prescaler
11010	IP Bus clock divide by 4 prescaler
11011	IP Bus clock divide by 8 prescaler
11100	IP Bus clock divide by 16 prescaler
11101	IP Bus clock divide by 32 prescaler
11110	IP Bus clock divide by 64 prescaler
11111	IP Bus clock divide by 128 prescaler

NOTE

A timer selecting its own output as its primary count source is not a legal choice. The result is no counting.

ONCE - Count Once

This bit selects continuous or one shot counting mode.

- 1 = Count until compare and then stop. When output mode \$4 is used, the counter re-initializes after reaching the COMP1 value and continues to count to the COMP2 value then stops.
- 0 = Count repeatedly.

LENGTH - Count Length

This bit determines whether the counter counts to the compare value and then re-initializes itself to the value specified in the LOAD, CMPLD1, or CMPLD2 registers, or the counter continues counting past the compare value, to the binary roll over.

- 1 = Count until compare, then reinitialize.

The value that the counter is reinitialized with depends on the settings of CLC1 and CLC2. If neither of these indicates the counter is to be loaded from one of the CMPLD registers, then the LOAD register is used to reinitialize the counter upon matching either COMP register. If one of CLC1 or CLC2 indicates that the counter is to be loaded from one of the CMPLD registers, then the counter will reinitialize to the value in the appropriate CMPLD register upon a match with the appropriate COMP register. If both of the CLC1 and CLC2 fields indicate that the counter is to be loaded from the CMPLD registers, then CMPLD1 will have priority if both compares happen at the same value.

When output mode \$4 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until COMP1 value is reached, re-initializes, then counts until COMP2 value is reached, re-initializes, then counts until COMP1 value is reached, etc.

- 0 = Continue counting to roll over.

DIR - Count Direction

This bit selects either the normal count direction *up*, or the reverse direction, *down*.

1 = Count down

0 = Count up

SECSRC - Secondary Count Source

These bits identify the source to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of the CNTR register. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the SIPS bit of the CTRL2 register.

Table 381. Secondary Count Source Values

Value	Meaning
00000	Counter #0 input pin
00001	Counter #1 input pin
00010	Counter #2 input pin
00011	Counter #3 input pin
00100	Counter #4 input pin
00101	Counter #5 input pin
00110	Counter #6 input pin
00111	Counter #7 input pin
01000	Reserved
01001	Reserved
01010	Auxiliary input #2 pin
01011	Reserved
01100	Reserved
01101	Reserved
01110	Reserved
01111	Reserved
10000	Counter #0 output
10001	Counter #1 output
10010	Counter #2 output
10011	Counter #3 output
10100	Counter #4 output
10101	Counter #5 output
10110	Counter #6 output
10111	Counter #7 output
11000 - 11111	Reserved

28.4.4.9 Control Register 2 (CTRL2)

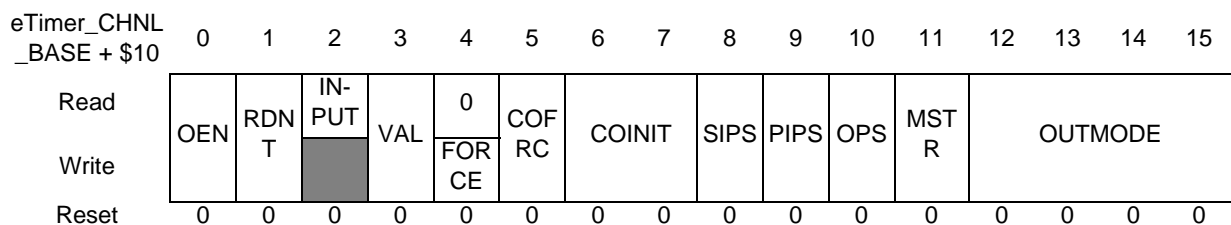


Figure 371. Control Register 2 (CTRL2)

OEN - Output Enable.

This bit determines the direction of the external pin.

1 = OFLAG output signal will be driven on the external pin. Other timer channels using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit.

0 = The external pin is configured as an input.

RDNT - Redundant Channel Enable.

This bit enables redundant channel checking between adjacent channels (0 and 1, 2 and 3, 4 and 5, 6 and 7). When this bit is clear, the RCF bit in this channel cannot be set. When this bit is set, the RCF bit will be set by a miscompare between the OFLAG of this channel and the OFLAG of its redundant adjacent channel which will cause the output of this channel to go inactive (logic 0 prior to consideration of the OPS bit).

1 = Enable redundant channel checking.

0 = Disable redundant channel checking.

INPUT - External input signal

This read only bit reflects the current state of the signal selected via SECSRC after application of the SIPS bit and filtering.

VAL - Forced OFLAG Value

This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs.

FORCE- Force the OFLAG output

This write only bit forces the current value of the VAL bit to be written to the OFLAG output. This bit always reads as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if OUTMODE is 0000 (software controlled). Setting this bit while the OUTMODE is a different value may yield unpredictable results.

COFRC - Co-channel OFLAG Force

This bit enables the compare from another channel within the module to force the state of this counter's OFLAG output signal.

1 = Other channels may force the OFLAG of this channel.

0 = Other channels cannot force the OFLAG of this channel.

COINIT - Co-channel Initialization

These bits enable another channel within the module to force the re-initialization of this channel when the other channel has an active compare event.

Table 382. Values for Co-channel Initialization

Value	Meaning
00	Other channels cannot force re-initialization of this channel.
01	Other channels may force a re-initialization of this channel's counter using the LOAD reg.
10	Other channels may force a re-initialization of this channel's counter with the CMPLD2 reg when this channel is counting down or the CMPLD1 reg when this channel is counting up.
11	Reserved.

SIPS - Secondary Source Input Polarity Select

This bit inverts the polarity of the signal selected by the SECSRC bits.

- 1 = Inverted polarity.
- 0 = True polarity.

PIPS - Primary Source Input Polarity Select

This bit inverts the polarity of the signal selected by the PRISRC bits. This only applies if the signal selected by PRISRC is not the prescaled IP Bus clock.

- 1 = Inverted polarity.
- 0 = True polarity.

OPS - Output Polarity Select.

This bit inverts the OFLAG output signal polarity.

- 1 = Inverted polarity.
- 0 = True polarity.

MSTR - Master Mode

This bit enables the compare function's output to be broadcasted to the other channels in the module. The compare signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs.

- 1 = Enable broadcast of compare events from this channel.
- 0 = Disable broadcast of compare events from this channel.

OUTMODE - Output Mode

These bits determine the mode of operation for the OFLAG output signal.

Table 383. OUTMODE Values

Value	Meaning
0000	Software controlled
0001	Clear OFLAG output on successful compare (COMP1 or COMP2)
0010	Set OFLAG output on successful compare (COMP1 or COMP2)
0011	Toggle OFLAG output on successful compare (COMP1 or COMP2)
0100	Toggle OFLAG output using alternating compare registers
0101	Set on compare with COMP1, cleared on secondary source input edge
0110	Set on compare with COMP2, cleared on secondary source input edge
0111	Set on compare, cleared on counter roll-over
1000	Set on successful compare on COMP1, clear on successful compare on COMP2
1001	Asserted while counter is active, cleared when counter is stopped.
1010	Asserted when counting up, cleared when counting down.
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Enable gated clock output while counter is active

28.4.4.10 Control Register 3 (CTRL3)

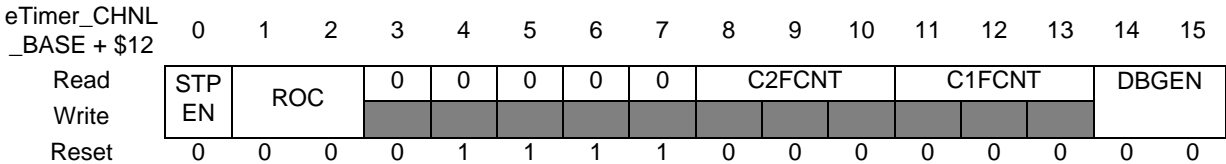


Figure 372. Control Register 3 (CTRL3)

STPEN - Stop Actions Enable

This bit allows the tri-stating of the timer output during stop mode.

- 1 = Output enable is disabled during stop mode.
- 0 = Output enable is unaffected by stop mode.

ROC - Reload on Capture

These bits enable the capture function to cause the counter to be reloaded from the LOAD register.

Table 384. Values for Reload on Capture

Value	Meaning
00	Do not reload the counter on a capture event.
01	Reload the counter on a capture 1 event.
10	Reload the counter on a capture 2 event.
11	Reload the counter on both a capture 1 event and a capture 2 event.

C2FCNT - CAPT2 FIFO Word Count

This field reflects the number of words in the CAPT2 FIFO.

C1FCNT - CAPT1 FIFO Word Count

This field reflects the number of words in the CAPT1 FIFO.

DBGEN - Debug Actions Enable

These bits allow the counter channel to perform certain actions in response to the chip entering debug mode.

Table 385. Values for DBGEN

Value	Meaning
00	Continue with normal operation during debug mode. (default)
01	Halt channel counter during debug mode.
10	Force OFLAG to logic 0 (prior to consideration of the OPS bit) during debug mode.
11	Both halt counter and force OFLAG to 0 during debug mode.

28.4.4.11 Status Register (STS)

eTimer_CHNL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
_BASE + \$14	0	0	0	0	0	0	WDF	RCF	ICF2	ICF1	IEHF	IELF	TOF	TCF 2	TCF 1	TCF
Read	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 373. Status Register (STS)

WDF - Watchdog Time-out Flag

This bit is set when the watchdog times out by counting down to zero. The watchdog must be enabled for time-out to occur and channel 0 must be in quadrature decode count mode (CNTMODE = 100).

This bit is cleared by writing a 1 to this bit position after either writing a non-zero value to WDTOL and/or WDTOH or exiting quadrature decode counting mode. This bit is only in channel 0.

RCF - Redundant Channel Flag

This bit is set when there is a miscompare between this channel's OFLAG value and the OFLAG value of the corresponding redundant channel. Corresponding channels are grouped together in the following pairs: 0 and 1, 2 and 3, 4 and 5, or 6 and 7. This bit can only be set if the RDNT bit is set. This bit is cleared by writing a 1 to this bit position. This bit is only in even channels (0, 2, 4, and 6).

ICF2 - Input Capture 2 Flag

This bit is set when an input capture event (as defined by CPT2MODE) occurs and the word count of the CAPT2 FIFO exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF2DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF2DE is set (DMA).

ICF1 - Input Capture 1 Flag

This bit is set when an input capture event (as defined by CPT1MODE) occurs and the word count of the CAPT1 FIFO exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF1DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF1DE is set (DMA).

IEHF - Input Edge High Flag

This bit is set when a positive input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.

IELF - Input Edge Low Flag

This bit is set when a negative input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.

TOF - Timer Overflow Flag

This bit is set when the counter rolls over its maximum value \$FFFF or \$0000 (depending on count direction). This bit is cleared by writing a one to this bit location.

TCF2 - Timer Compare 2 Flag

This bit is set when a successful compare occurs with COMP2. This bit is cleared by writing a one to this bit location.

TCF1 - Timer Compare 1 Flag

This bit is set when a successful compare occurs with COMP1. This bit is cleared by writing a one to this bit location.

TCF - Timer Compare Flag

This bit is set when a successful compare occurs. This bit is cleared by writing a one to this bit location.

28.4.4.12 Interrupt and DMA Enable Register (INTDMA)

eTimer_CHNL _BASE + \$16	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ICF2 DE	ICF1 DE	CMP LD2 DE	CMP LD1 DE	0	0	WDF	RCF	ICF2	ICF1	IEHF	IELF	TOF	TCF	TCF	TCF
Write	DE	DE	DE	DE			IE	IE	IE	IE	IE	IE	IE	2IE	1IE	IE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 374. Interrupt and DMA Enable Register (INTDMA)

ICF2DE - Input Capture 2 Flag DMA Enable

Setting this bit enables DMA read requests for CAPT2 when the ICF2 bit is set. Do not set both this bit and the ICF2IE bit.

ICF1DE - Input Capture 1 Flag DMA Enable

Setting this bit enables DMA read requests for CAPT1 when the ICF1 bit is set. Do not set both this bit and the ICF1IE bit.

CMPLD2DE - Comparator Load Register 2 Flag DMA Enable

Setting this bit enables DMA write requests to the CMPLD2 register whenever data is transferred out of the CMPLD2 reg into either the CNTR, COMP1, or COMP2 registers.

CMPLD1DE - Comparator Load Register 1 Flag DMA Enable

Setting this bit enables DMA write requests to the CMPLD1 register whenever data is transferred out of the CMPLD1 reg into either the CNTR, COMP1, or COMP2 registers.

WDFIE - Watchdog Flag Interrupt Enable

Setting this bit enables interrupts when the WDF bit is set. This bit is only in channel 0.

RCFIE - Redundant Channel Flag Interrupt Enable

Setting this bit enables interrupts when the RCF bit is set. This bit is only in even channels (0, 2, 4, and 6).

ICF2IE - Input Capture 2 Flag Interrupt Enable

Setting this bit enables interrupts when the ICF2 bit is set. Do not set both this bit and the ICF2DE bit.

ICF1IE - Input Capture 1 Flag Interrupt Enable

Setting this bit enables interrupts when the ICF1 bit is set. Do not set both this bit and the ICF1DE bit.

IEHFIE - Input Edge High Flag Interrupt Enable

Setting this bit enables interrupts when the IEHF bit is set.

IELFIE - Input Edge Low Flag Interrupt Enable

Setting this bit enables interrupts when the IELF bit is set.

TOFIE - Timer Overflow Flag Interrupt Enable

Setting this bit enables interrupts when the TOF bit is set.

TCF2IE - Timer Compare 2 Flag Interrupt Enable

Setting this bit enables interrupts when the TCF2 bit is set.

TCF1IE - Timer Compare 1 Flag Interrupt Enable

Setting this bit enables interrupts when the TCF1 bit is set.

TCFIE - Timer Compare Flag Interrupt Enable

Setting this bit enables interrupts when the TCF bit is set.

28.4.4.13 Comparator Load Register 1 (CMPLD1)

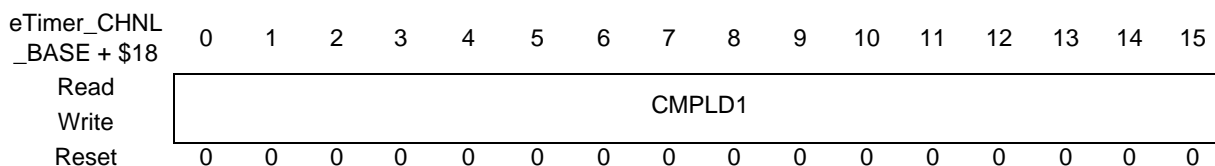


Figure 375. Comparator Load 1 (CMPLD1)

This read/write register is the preload value for the COMP1 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in [Section 28.3.2.14, “Usage of Compare Load Registers”](#).

28.4.4.14 Comparator Load Register 2 (CMPLD2)

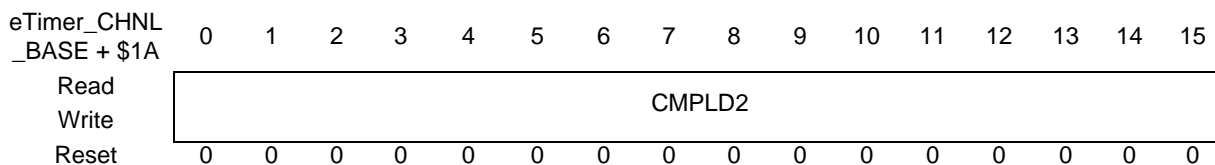


Figure 376. Comparator Load 2 (CMPLD2)

This read/write register is the preload value for the COMP2 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in [Section 28.3.2.14, “Usage of Compare Load Registers”](#).

28.4.4.15 Compare and Capture Control Register (CCCTRL)

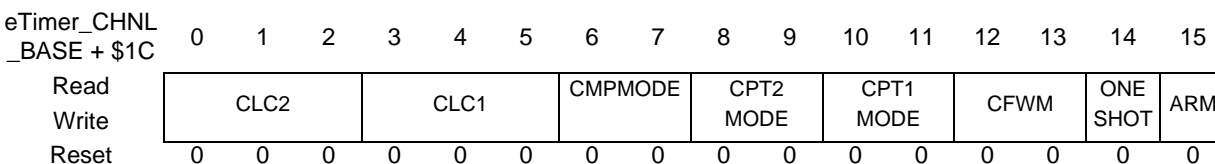


Figure 377. Compare and Capture Control Register (CCCTRL)

CLC2 - Compare Load Control 2

These bits control when COMP2 is preloaded. It also controls the loading of CNTR.

Table 386. Values for Compare Load Control 2

Value	Meaning
000	Never preload
001	Reserved
010	Load COMP2 with CMPLD1 upon successful compare with the value in COMP1.
011	Load COMP2 with CMPLD1 upon successful compare with the value in COMP2.
100	Load COMP2 with CMPLD2 upon successful compare with the value in COMP1.
101	Load COMP2 with CMPLD2 upon successful compare with the value in COMP2.
110	Load CNTR with CMPLD2 upon successful compare with the value in COMP1.
111	Load CNTR with CMPLD2 upon successful compare with the value in COMP2.

CLC1 - Compare Load Control 1

These bits control when COMP1 is preloaded. It also controls the loading of CNTR.

Table 387. Values for Compare Load Control 1

Value	Meaning
000	Never preload
001	Reserved
010	Load COMP1 with CMPLD1 upon successful compare with the value in COMP1.
011	Load COMP1 with CMPLD1 upon successful compare with the value in COMP2.
100	Load COMP1 with CMPLD2 upon successful compare with the value in COMP1.
101	Load COMP1 with CMPLD2 upon successful compare with the value in COMP2.
110	Load CNTR with CMPLD1 upon successful compare with the value in COMP1.
111	Load CNTR with CMPLD1 upon successful compare with the value in COMP2.

CMPMODE - Compare Mode

These bits control when the COMP1 and COMP2 registers are used in regards to the counting direction.

Table 388. Values for Compare Mode

Value	Meaning
00	COMP1 register is used when the counter is counting up. COMP2 register is used when the counter is counting up.
01	COMP1 register is used when the counter is counting down. COMP2 register is used when the counter is counting up.
10	COMP1 register is used when the counter is counting up. COMP2 register is used when the counter is counting down.
11	COMP1 register is used when the counter is counting down. COMP2 register is used when the counter is counting down.

CPT2MODE - Capture 2 Mode Control

These bits control the operation of the CAPT2 register as well as the operation of the ICF2 flag by defining which input edges cause a capture event. The input source is the secondary count source.

Table 389. Values for Capture 2 Mode Control

Value	Meaning
00	Disabled
01	Capture falling edges
10	Capture rising edges.
11	Capture any edge.

CPT1MODE - Capture 1 Mode Control

These bits control the operation of the CAPT1 register as well as the operation of the ICF1 flag by defining which input edges cause a capture event. The input source is the secondary count source.

Table 390. Values for Capture 1 Mode Control

Value	Meaning
00	Disabled
01	Capture falling edges
10	Capture rising edges.
11	Capture any edge.

CFWM - Capture FIFO Water Mark

This field represents the water mark level for the CAPT1 and CAPT2 FIFOs. The capture flags, ICF1 and ICF2, won't be set until the word count of the corresponding FIFO is greater than this water mark level.

ONESHOT - One Shot Capture Mode

This bit selects between free running and one shot mode for the input capture circuitry.

1 = One shot mode is selected.

If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and the ARM bit is cleared. No further captures will be performed until the ARM bit is set again.

If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARM bit is then cleared.

0 = Free running mode is selected

If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and capture circuit 1 is re-armed. The process continues indefinitely.

If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit.

ARM - Arm Capture

Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s).

- 1 = Input capture operation as specified by the CPT1MODE and CPT2MODE bits is enabled.
- 0 = Input capture operation is disabled.

28.4.4.16 Input Filter Register (FILT)

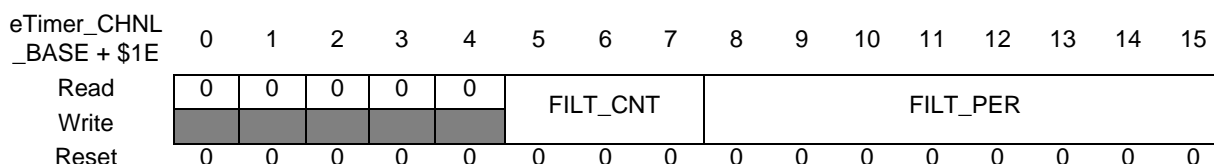


Figure 378. Input Filter Register (FILT)

FILT_CNT - Input Filter Sample Count

These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in [Section 28.4.4.16.1, “Input Filter Considerations”](#).

FILT_PER - Input Filter Sample Period

These bits represent the sampling period (in IPBus clock cycles) of the eTimer input signal. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is \$00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in [Section 28.4.4.16.1, “Input Filter Considerations”](#).

28.4.4.16.1 Input Filter Considerations

The FILT_PER value should be set such that the sampling period is larger the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of: $((\text{FILT_CNT} + 3) \times \text{FILT_PER}) + 2$ IPBus clock periods.

28.4.5 Watchdog Timer Registers

The base address of the Watchdog Timer registers is equal to the base address of the eTimer plus an offset of \$100.

28.4.5.1 Watchdog Time-out Registers (WDTOL and WDTOH)

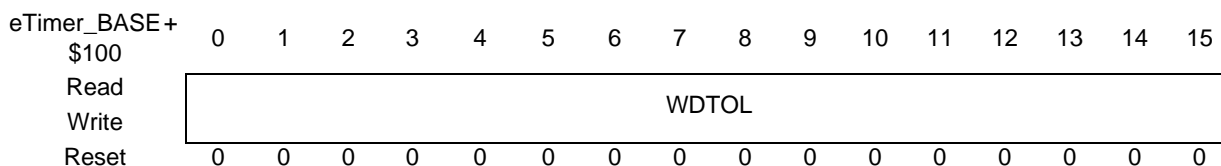


Figure 379. Watchdog Time-out Low Word Register (WDTOL)

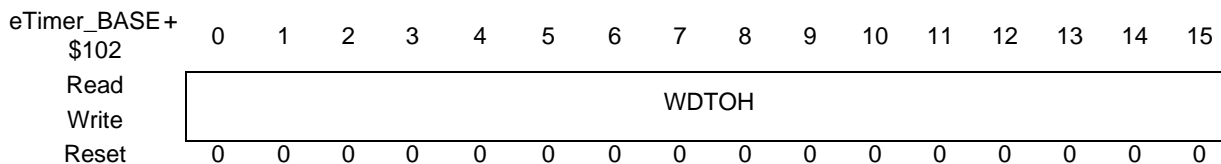


Figure 380. Watchdog Time-out High Word Register (WDTOH)

WDTO - Watchdog Time-out

These registers are combined to form the 32 bit time-out count for the Timer watchdog function. This time-out count is used to monitor for inactivity on the inputs when channel 0 is in the quadrature decode count mode. The watchdog function is enabled whenever WDTO contains a non-zero value (although actual counting only occurs if channel 0 is in quadrature decode counting mode). The watchdog time-out down counter is loaded whenever WDTOH is written. These registers are not byte accessible. See Section 28.3.3.5, “Watchdog Timer”, for more information on the use of the watchdog timer.

28.4.6 Configuration Registers

The base address of the configuration registers is equal to the base address of the eTimer plus an offset of \$10C.

28.4.6.1 Channel Enable Register (ENBL)

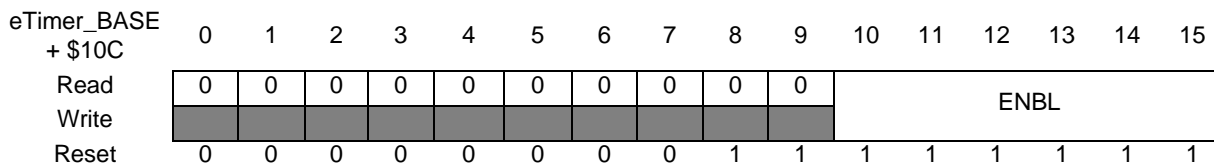


Figure 381. Channel Enable Register (ENBL)

ENBL - Timer Channel Enable

These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate channels. If an ENBL bit is set, then the corresponding channel will start counting as soon as the CNTMODE field has a value other than 000. When an ENBL bit is clear, the corresponding channel maintains its current value.

- 1 = Timer channel is enabled. (default)
- 0 = Timer channel is disabled.

28.4.6.2 DMA Request Select Registers (DREQ0, DREQ1)

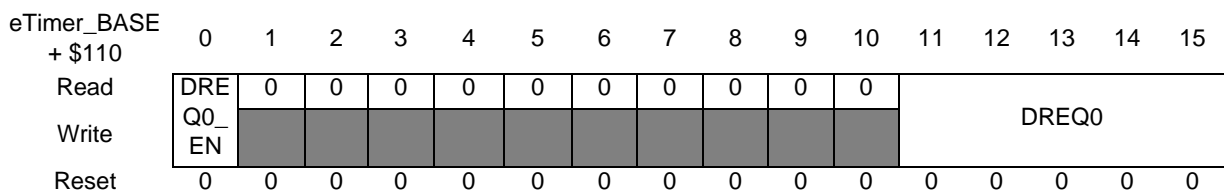


Figure 382. DMA Request 0 Select Register (DREQ0)

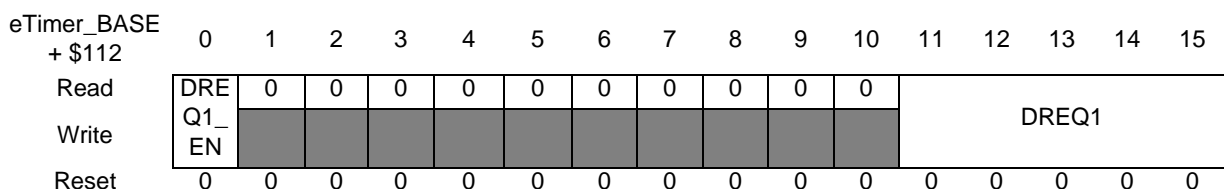


Figure 383. DMA Request 1 Select Register (DREQ1)

DREQn_EN - DMA Request Enable

Use these bits to enable each of the four module level DMA request outputs. Program the DREQ fields prior to setting the corresponding enable bit. Clearing this enable bit will remove the request but won't clear the flag that is causing the request.

- 1 = DMA request enabled.
- 0 = DMA request disabled.

DREQn - DMA Request Select

Use these fields to select which DMA request source will be muxed onto one of the four module level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed.

Table 391. Values for DREQn

Value	Selected DMA Request
00000	Channel 0 CAPT1 DMA read request
00001	Channel 0 CAPT2 DMA read request
00010	Channel 0 CMPLD1 DMA write request
00011	Channel 0 CMPLD2 DMA write request
00100	Channel 1 CAPT1 DMA read request
00101	Channel 1 CAPT2 DMA read request
00110	Channel 1 CMPLD1 DMA write request
00111	Channel 1 CMPLD2 DMA write request
01000	Channel 2 CAPT1 DMA read request
01001	Channel 2 CAPT2 DMA read request
01010	Channel 2 CMPLD1 DMA write request

Table 391. Values for DREQn (continued)

Value	Selected DMA Request
01011	Channel 2 CMPLD2 DMA write request
01100	Channel 3 CAPT1 DMA read request
01101	Channel 3 CAPT2 DMA read request
01110	Channel 3 CMPLD1 DMA write request
01111	Channel 3 CMPLD2 DMA write request
10000	Channel 4 CAPT1 DMA read request
10001	Channel 4 CAPT2 DMA read request
10010	Channel 4 CMPLD1 DMA write request
10011	Channel 4 CMPLD2 DMA write request
10100	Channel 5 CAPT1 DMA read request
10101	Channel 5 CAPT2 DMA read request
10110	Channel 5 CMPLD1 DMA write request
10111	Channel 5 CMPLD2 DMA write request
11000	Channel 6 CAPT1 DMA read request
11001	Channel 6 CAPT2 DMA read request
11010	Channel 6 CMPLD1 DMA write request
11011	Channel 6 CMPLD2 DMA write request
11100	Channel 7 CAPT1 DMA read request
11101	Channel 7 CAPT2 DMA read request
11110	Channel 7 CMPLD1 DMA write request
11111	Channel 7 CMPLD2 DMA write request

28.5 Resets

The eTimer module can only be reset by the `IPG_HARD_ASYNC_RESET_B` signal. This forces all registers to their reset state and clears the `OFLAG` signals if they are asserted. The counters will be turned off until the settings in the `CTRL` registers are changed.

Table 392. Reset Summary

Reset	Source	Characteristics
<code>IPG_HARD_ASYNC_RESET_B</code>	Hardware Reset	Full System Reset

28.6 Clocks

Each timer channel receives its own copy of the IPBus Clock. This allows for better power management by turning off the clock to unused channels. The watchdog timer has its own version of the IPBus clock. Each bit of the `ENBL` register uses the clock for that corresponding channel. The `DREQn` registers use the clock for channel 0 but this clock can be turned off after these registers have been programmed.

Table 393. Clock Summary

Clock	Used by
ipg_clk_ch0	Channel 0 and DREQn
ipg_clk_ch1	Channel 1
ipg_clk_ch2	Channel 2
ipg_clk_ch3	Channel 3
ipg_clk_ch4	Channel 4
ipg_clk_ch5	Channel 5
ipg_clk_wdog	Watchdog

28.7 Interrupts

Each of the channels within the eTimer can generate an interrupt from several sources. The watchdog also generates interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 394. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
Channels 0-5	TCF	TCFIE	Compare interrupt	Compare of counter and related compare register
	TCF1	TCF1IE	Compare 1 interrupt	Compare of the counter and COMP1 register
	TCF2	TCF2IE	Compare 2 interrupt	Compare of the counter and COMP2 register
	TOF	TOFIE	Overflow interrupt	Generated on counter roll-over or roll-under
	IELF	IELFIE	Input Low Edge interrupt	Falling edge of the secondary input signal
	IEHF	IEHFIE	Input High Edge interrupt	Rising edge of the secondary input signal
	ICF1	ICF1IE	Input Capture 1 interrupt	Input capture event for CAPT1
	ICF2	ICF2IE	Input Capture 2 interrupt	Input capture event for CAPT2
Watchdog	WDF	WDFIE	Watchdog time-out interrupt	Watchdog has timed out
Redundant Channel Checking	RCF	RCFIE	Redundant Channel Fault interrupt	Miscompare with redundant channel

28.8 DMA

Each channel can request a DMA read access for each of the capture registers and a DMA write request for each of the compare preload registers. The DREQ registers select amongst these 24 DMA request sources to generate the 2 top level DMA request outputs.

When DMA is used to read the eTimer_CAPTn registers and the DMA has completed its programmed number of transfers, then the extra input capture events will set the eTimer_STS[ICFn] bits and also set the internal DMA request signal. While the ICFn bits can be cleared by writing a 1 to their bit positions,

the DMA request can only be cleared by the internal DMA done signal. This means that when a new DMA transfer is programmed, the eTimer will request a DMA read with possibly unwanted data. In cases where extra eTimer input capture events might occur, the following procedure can be used to prevent unwanted DMA read requests: Upon completion of the DMA transfer, clear the Timer_INTDMA[ICFnDE] bits.

Table 395. DMA Summary

DMA Request	DMA Enable	Name	Description
Channels 0-5	ICF1DE	CAPT1 read request	CAPT1 contains a value
	ICF2DE	CAPT2 read request	CAPT2 contains a value
	CMPLD1DE	CMPLD1 write request	CMPLD1 needs an update
	CMPLD2DE	CMPLD2 write request	CMPLD2 needs an update

28.9 ADC Trigger

eTimer module can also be used to trigger ADC. See [Section 27.4.1.4, “Injected channel conversion”](#) for details.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 29

Fault Collection Unit (FCU)

29.1 Introduction

The Fault Collection Unit (FCU) module provides functional safety to the device.

29.1.1 Overview

The FCU provides a central capability to collect faults reported by the individual modules of the device. It represents the minimum blocking unit to develop a coherent safety strategy for the chassis family. Selected critical faults are reported to the external device via output pins, if no recovery is provided by the device. The operation of the FCU is independent from the CPU. The FCU provides an independent fault reporting mechanism even in case the CPU behavior is abnormal. The FCU always starts up in init mode. As long as the FCU remains in init mode, testing of the FCU logic (for dormant fault detection) can be performed under software control.

The FCU is developed to increase the level of safety of the system/MCU level.

Functional safety features of the FCU include:

- It is an independent module: If other control modules are behaving abnormally, the user can still trigger actions to prevent a critical situation.
- Collection and external reporting of faults occurring on the device
- Centralized fault collection
- Each fault cause can be treated in a different way
 - No action
 - Alarm—allows hardware or software to recover from fault
 - Fault—communicates directly to an external device that something went wrong
- Three different output protocols available
- Possible to inject fake faults on user request during initialization phase to test the peripheral (dormant fault detection)

29.1.1.1 General description

The FCU is logically divided in three blocks:

- Input unit—captures faults reported from the device
- Control unit—implemented by a finite state machine (see [Figure 398](#) for details)
- Output unit—generates output signals

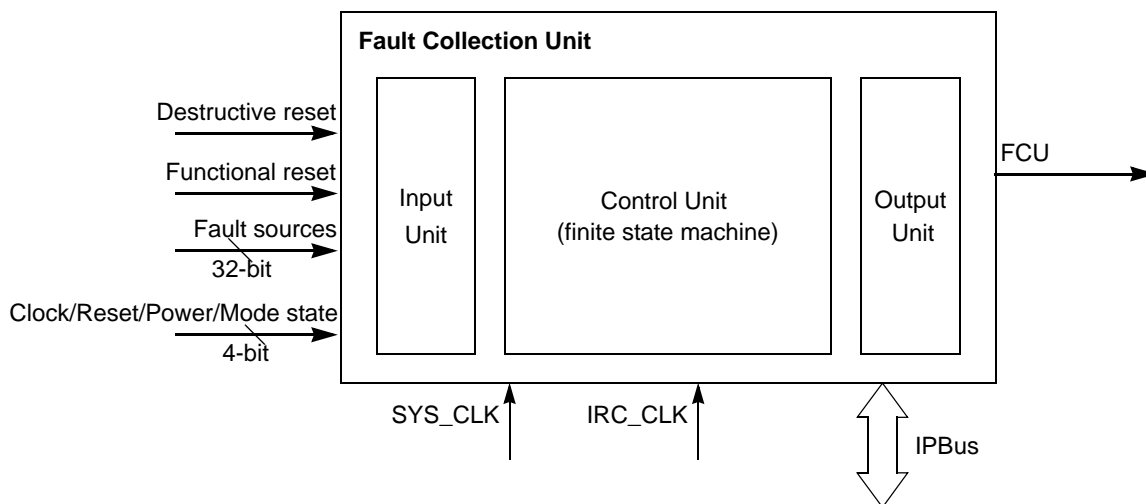


Figure 384. Fault Collection Unit (FCU) block diagram

Figure 385 shows the flow chart of FCU fault handling.

The FCU module and its state machine run on the system clock, making the two modules synchronous. The high speed RC clock (16 MHz) is used only in the Alarm state, in order to compute a deterministic timeout.

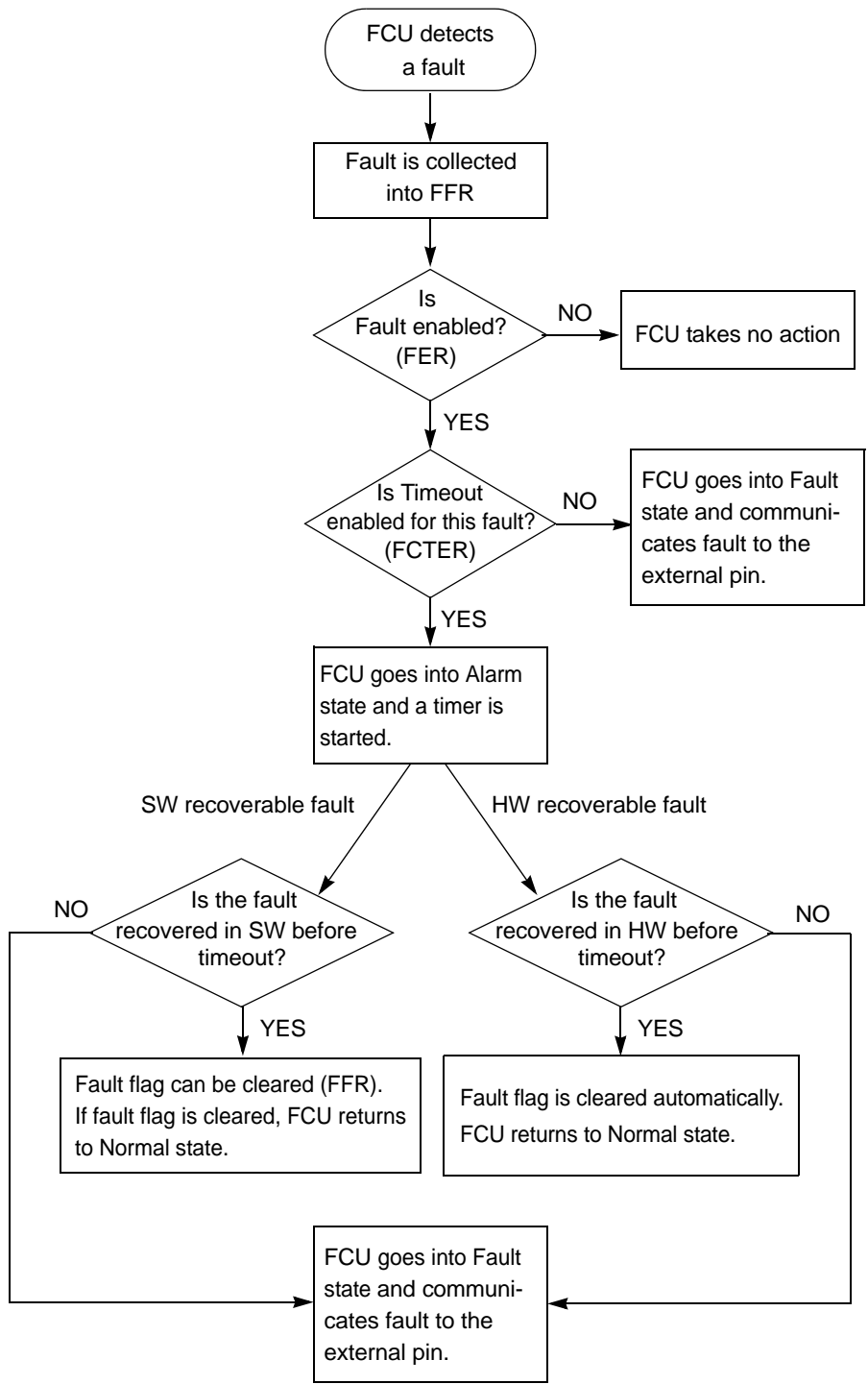


Figure 385. FCU fault handling

29.1.2 Features

The FCU includes the following features:

- Collection of critical faults
- Reporting of selected critical faults to external pins
- Fault flag status kept over non-destructive reset for later analysis (in a “Freeze” register)
- Continuous and synchronous latch of MC state
- MC state kept over non-destructive reset for later analysis (in a “Freeze” register)
- 4-state (Init, Normal, Alarm, Fault) finite state machine
- Different actions can be taken depending on fault type
- Selectable protocols for fault signal indication in Fault state (dual-rail, time-switching, bi-stable)
- Programmable clock prescaler for time-switching output signal generation
- Protection mechanism to avoid unwanted clearing of fault flags
- Internal logic testing, by using a fake fault generator during initialization phase

29.1.3 Modes of operation

This section describes the basic functional modes of the FCU module.

29.1.3.1 Normal mode

In Normal operation, the FCU captures all the faults in real time and processes them according to the fault type and the configuration set by the user.

29.1.3.2 Test mode

Test mode provides a testing mechanism of the FCU (for dormant fault detection). Test mode can be entered during the configuration (Init) phase. The user can write to the Fake Fault Generation Register (FCU_FFGR) and can check the behavior while staying in Test mode. During Test mode, the state machine behaves normally. In Test mode, real faults are not detected.

The user can inject fake faults by writing to the FCU_FFGR during Test mode to test the peripheral. A software-triggered fault is not supported by the FCU_FFGR; it will be taken from Fault input even in Test mode. Fault flags are cleared when Test mode is exited. Asynchronous software faults written to the FCU_FFGR during Init phase will not be latched. In order to test software faults, the FCU_FFGR needs to be cleared during Init phase and written to during Normal phase.

29.2 Memory map and register definition

The following sections define the FCU memory map, register layout and functionality.

29.2.1 Memory map

Table 396. FCU memory map

Offset from FCU_BASE (0xFFE6_C000)	Register	Access	Reset value	Location
0x0000	Module Configuration Register (FCU_MCR)	R/W	0x0000_0007	on page 731
0x0004	Fault Flag Register (FCU_FFR)	R/W	0x0000_0000	on page 732
0x0008	Frozen Fault Flag Register (FCU_FFFR)	R	0x0000_0000	on page 734
0x000C	Fake Fault Generation Register (FCU_FFGR)	R/W	0x0000_0000	on page 735
0x0010	Fault Enable Register (FCU_FER)	R/W	0xF800_FFFF	on page 736
0x0014	Key Register (FCU_KR)	W	0x0000_0000	on page 736
0x0018	Timeout Register (FCU_TR)	R/W	0x0000_FFFF	on page 737
0x001C	Timeout Enable Register (FCU_TER)	R/W	0x0000_0000	on page 738
0x0020	Module State Register (FCU_MSR)	R	0x0000_0001	on page 738
0x0024	Microcontroller State Register (FCU_MCSR)	R	0x0000_0003	on page 739
0x0028	Frozen MC State Register (FCU_FMCSR)	R	0x0000_0000	on page 740
0x002C–0x3FFF	Reserved			

29.2.2 Register summary

Table 397 shows the register summary.

Table 397. Register summary

Name	0		1		2		3		27		5		6		7		8		9		10		11		12		13		14		15	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
0x0000_0000 FCU_MCR	R	MCL		TM[1:0]		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																															
	R	0	0	0	0	0	0	PS[1:0]		FOM[1:0]		FOP[5:0]																				
	W																															
0x0000_0004 FCU_FFR	R	0	0	SRF 2	SRF 3	SRF 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	W																															
	R	HRF 15	HRF 14	HRF 13	HRF 12	HRF 11	HRF 10	HRF 9	HRF 8	HRF 7	HRF 6	HRF 5	HRF 4	HRF 3	HRF 2	HRF 1	HRF 0															
	W																															

Table 397. Register summary (continued)

Name		0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0000_0008 FCU_FFR	R	0	0	FR SRF 2	FR SRF 3	FR SRF 4	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	FR HRF 15	FR HRF 14	FR HRF 13	FR HRF 12	FR HRF 11	FR HRF 10	FR HRF 9	FR HRF 8	FR HRF 7	FR HRF 6	FR HRF 5	FR HRF 4	FR HRF 3	FR HRF 2	FR HRF 1	FR HRF 0
	W																
0x0000_000C FCU_FFGR	R	0	0	FSR F2	FSR F3	FSR F4	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	F HRF 15	F HRF 14	F HRF 13	F HRF 12	F HRF 11	F HRF 10	F HRF 9	F HRF 8	F HRF 7	F HRF 6	F HRF 5	F HRF 4	F HRF 3	F HRF 2	F HRF 1	F HRF 0
	W																
0x0000_0010 FCU_FER	R	0	0	ESF2	ESF3	ESF4	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EHF 15	EHF 14	EHF 13	EHF 12	EHF 11	EHF 10	EHF 9	EHF 8	EHF 7	EHF 6	EHF 5	EHF 4	EHF 3	EHF 2	EHF 1	EHF 0
	W																
0x0000_0014 FCU_KR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x0000_0018 FCU_TR	R	TR[31:16]															
	W																
	R	TR[15:0]															
	W																
0x0000_001C FCU_TER	R	0	0	TES F2	TES F3	TES F4	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TEH F15	TEH F14	TEH F13	TEH F12	TEH F11	TEH F10	TEH F9	TEH F8	TEH F7	TEH F6	TEH F5	TEH F4	TEH F3	TEH F2	TEH F1	TEH F0
	W																
0x0000_0020 FCU_MSR	R	0	0	0	0	0	0	0	0	0	0	0	0	S0	S1	S2	S3
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	S0	S1	S2	S3
	W																

Table 397. Register summary (continued)

Name	0		1		2		3		27		5		6		7		8		9		10		11		12		13		14		15	
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
0x0000_0024 FCU_MCSR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MCPS[3:0]															
	W																															
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MCAS[3:0]															
	W																															
0x0000_0028 FCU_FMCSR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FRMCPS[3:0]																
	W																															
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FRMCAS[3:0]																
	W																															

29.2.3 Register descriptions

29.2.3.1 Module Configuration Register (FCU_MCR)

The FCU_MCR does the following:

- Locks the configuration and lets the FCU go into Normal behavior state
- Enters Test mode (only possible during the Init state) but can be exited during any phase
- Configures protocol, prescaler, and polarity for FCU output signals (only possible before configuration is locked)

In Test mode, the FCU_FFGR can be accessed to emulate software/hardware faults. Fake faults can be generated only when Test mode is entered.

In Test mode, output pin(s) can be enabled or disabled, depending on the value of field TM[1:0]. To exit from Test mode, field TM must be written either '00' or '11'. While exiting the Test mode, the FCU must return to the Init state and automatically clear all the fault flags.

Address: 0x0000 Access: User read/write, Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MCL	TM[1:0]		0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	PS[1:0]		FOM[1:0]		FOP[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 386. Module Configuration Register (FCU_MCR)

Table 398. FCU_MCR field description

Field	Description
MCL	Module Configuration Lock 0: Configuration not locked, FCU remains in Init state 1: Configuration locked, FCU moves to Normal state
TM[1:0]	Test Mode 00: Test Mode not entered 01: Test Mode entered (fake faults can be generated), output pins disabled 10: Test Mode entered (fake faults can be generated), output pins enabled 11: Test Mode not entered
PS[1:0]	Polarity select 00: FCU[0] has normal polarity, FCU[1] has normal polarity. 01: FCU[0] has inverted polarity, FCU[1] has normal polarity. 10: FCU[0] has normal polarity, FCU[1] has inverted polarity. 11: FCU[0] has inverted polarity, FCU[1] has inverted polarity.
FOM[1:0]	Fault output mode selection 00: Dual-Rail (default state) 01: Time Switching 10: Bi-Stable 11: Reserved
FOP[5:0]	Fault Output Prescaler 000000: Input clock frequency is divided by $[4096 \times (0 + 1) \times 2]$, where 4096 is a fixed prescaler. 000001: Input clock frequency is divided by $[4096 \times (1 + 1) \times 2]$, where 4096 is a fixed prescaler. 000010: Input clock frequency is divided by $[4096 \times (2 + 1) \times 2]$, where 4096 is a fixed prescaler. 000011: Input clock frequency is divided by $[4096 \times (3 + 1) \times 2]$, where 4096 is a fixed prescaler. 000100: Input clock frequency is divided by $[4096 \times (4 + 1) \times 2]$, where 4096 is a fixed prescaler. ... Default at reset is 0x07 (input clock frequency is divided by $[4096 \times (7 + 1) \times 2]$).

29.2.3.2 Fault Flag Register (FCU_FFR)

The FCU_FFR contains the latched fault indication coming from the device. The FCU reacts on faults only if the respective enable bit for a fault is set in the Fault Enable Register (FCU_FER). In this case, the Alarm or Fault state is entered, depending on the user's selection. To enter Alarm state, the bit for the fault has to be set in the Timeout Enable Register (FCU_TER), otherwise Fault state is entered and output pins are communicating the internal fault.

No faults are latched in Init state (refer to state machine [Figure 398](#)).

The FCU_FFR is copied into Frozen Fault Flag Register (FCU_FFFR) only when the FCU goes into Fault state.

Each single flag can be cleared:

- In hardware, if the flag disappears due to hardware intervention. Bits 16:31 of the FCU_FFR store hardware-recoverable flags.
- In software, if the user application can recover from a faulty situation. Bits 0:4 of the FCU_FFR store software-recoverable flags.

Notice that software recoverable fault flags must be kept high also if the relative signal does not show anymore a fault. Hardware recoverable fault flags are updated in real time. Reset requests are assumed as hardware-recoverable.

In order to clear a flag in the FCU_FFR via software, the application should access first time the Key Register (FCU_KR) by writing the value of a key (0x618B_7A50) second time resetting the appropriate flag. In order to clear the software fault flag SRF1 the same protection mechanism has to be followed. The following errors are ignored:

- Wrong key inserted in FCU_KR
- Attempt to clear a hardware recoverable flag

If user software tries to clear an already cleared software-recoverable flag, SRF0 is set.

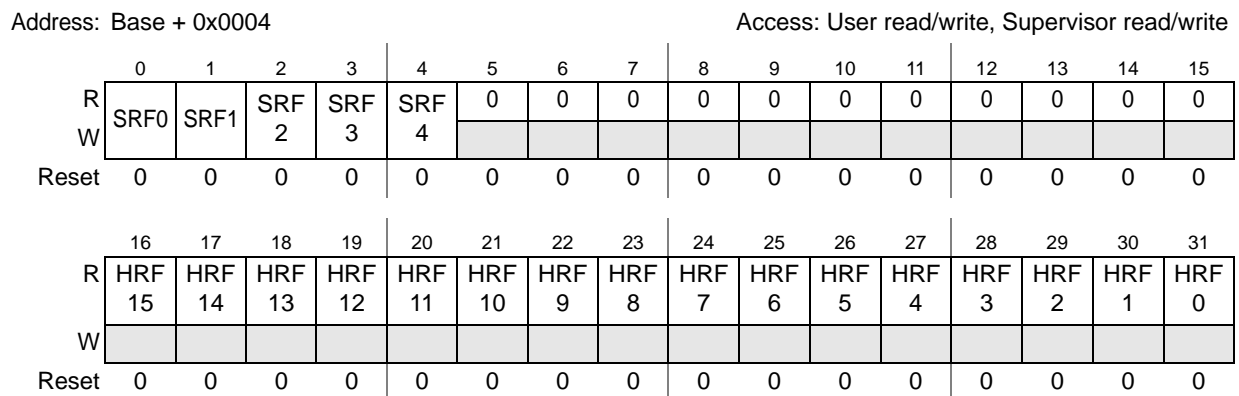


Figure 387. Fault Flag Register (FCU_FFR)

In addition to the detailed field descriptions in [Table 399](#), [Table 400](#) provides the hardware/software fault descriptions.

Table 399. FCU_FFR field descriptions

Field	Description
SRF0–SRF4	Software Recoverable Fault [2:15] 0: No error latched 1: Error latched
HRF15–HRF0	Hardware Recoverable Fault [15:0] 0: No error latched 1: Error latched

Table 400. Hardware/software fault description

Label	Module	Fault type
HRF0	Core	Checkstop mode entered
HRF1	Core	z0h core reset output
HRF2	CMU_0	Loss of crystal
HRF3	FMPLL_0	Loss of lock

Table 400. Hardware/software fault description (continued)

Label	Module	Fault type
HRF4	CMU_0	Frequency out of range
HRF5		Not used
HRF6		Not used
HRF7	Flash	Flash Fatal Error
HRF8		Not used
HRF9	JTAG	JTAG reset (TAP controller)
HRF10		Not used
HRF11		Not used
HRF12		Not used
HRF13		Not used
HRF14		Not used
HRF15		Not used
SRF0		Not used
SRF1		Not used
SRF2	Code Flash	ECC multi-bit error
SRF3	Data Flash	ECC multi-bit error
SRF4	SRAM	ECC multi-bit error

29.2.3.3 Frozen Fault Flag Register (FCU_FFFR)

The Fault Flag Register (FCU_FFR) is copied into Frozen Fault Flag Register (FFFR) every time the FCU goes into Fault state, in order to take a snapshot of the fault flags.

The bit description of the FCU_FFFR is the same as that of the FCU_FFR. The FCU_FFR is read/clear whereas the FCU_FFFR is read-only. See [Figure 388](#) and [Table 401](#) for details.

Address: Base + 0x0008 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FRS RF2	FRS RF3	FRS RF4	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FRH RF15	FRH RF14	FRH RF13	FRH RF12	FRH RF11	FRH RF10	FRH RF9	FRH RF8	FRH RF7	FRH RF6	FRH RF5	FRH RF4	FRH RF3	FRH RF2	FRH RF1	FRH RF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 388. Frozen Fault Flag Register (FCU_FFFR)

Table 401 provides a detailed bit description. Table 400 provides a detailed list of recoverable faults.

NOTE

If a fault source is configured as reset and FCU timeout feature is enabled, RGM_FES should be read to determine the fault source and not FCU_FFFR.

Table 401. FCU_FFFR field descriptions

Field	Description
FRSRF2– FRSRF4	Software Recoverable Fault 0: No error latched 1: Error latched
FRHRF15– FRHRF0	Hardware Recoverable Fault 0: No error latched 1: Error latched

29.2.3.4 Fake Fault Generation Register (FCU_FFGR)

The FCU_FFGR allows the user to emulate a software/hardware recoverable fault in order to test the FCU logic. Once a bit in the FCU_FFGR is set, the FCU should behave exactly in the same way after detecting a real fault. This register can be accessed only when Test mode is entered (check TM field in FCU_MCR). In Test mode, real faults are not detected and fake faults for SRF0 and SRF1 cannot be generated.

Address: Base + 0x000C

Access: User read/write, Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FSRF	FSRF	FSRF	0	0	0	0	0	0	0	0	0	0	0
W			2	3	4											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF	FHRF
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 389. Fake Fault Generation Register (FCU_FFGR)

Table 402. FCU_FFGR field description

Field	Description
FSRF2– FSRF4	Fake Software Recoverable Fault[2:4] 0: No error latched 1: Error latched
FHRF15– FHRF0	Fake Hardware Recoverable Fault[15:0] 0: No error latched 1: Error latched

29.2.3.5 Fault Enable Register (FCU_FER)

When a fault occurs, the FCU goes into either Alarm or Fault state (state is selected in the FCU_TER), if the respective fault enable bit is set in the Fault Enable Register (FCU_FER). This register can be configured only during the Init phase before the configuration is locked.

Address: Base + 0x0010 Access: User read/write, Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF	EHF
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 390. Fault Enable Register (FCU_FER)

Table 403. FCU_FER field descriptions

Field	Description
ESF2–ESF4	Enable Software Recoverable Fault 0: FCU takes no action on Software recoverable Fault [2:4] 1: FCU goes into Alarm/Fault state on Software recoverable Fault [2:4]
EHF15–EHF0	Enable Hardware Recoverable Fault 0: FCU takes no action on Hardware recoverable Fault [15:0] 1: FCU goes into Alarm/Fault state on Hardware recoverable Fault [15:0]

29.2.3.6 Key Register (FCU_KR)

The FCU_KR register implements the key access to clear the status flags of the FCU_FFR register.

The status bits of the FCU_FFR register, configured as software recoverable faults, can be cleared by the following locked sequence:

1. Write the FK key into the FCU_KR register.
2. Clear the status (flag) bit FCU_FFR.

NOTE

The FCU_KR register is not readable. 0x0000_0000 is always returned in case of a read operation.

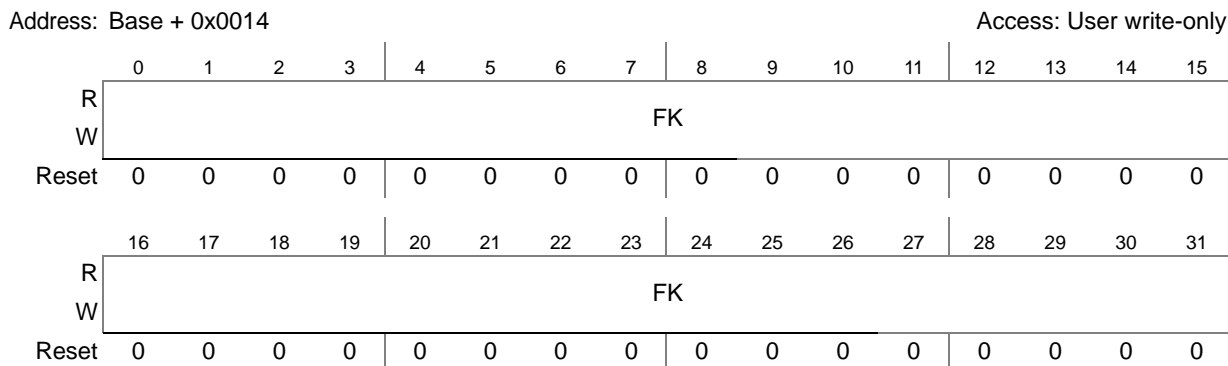


Figure 391. Key Register (FCU_KR)

Table 404. FCU_KR field descriptions

Field	Description
FK	Fault Key = 0x618B_7A50

29.2.3.7 Timeout Register (FCU_TR)

Once the FCU goes into Alarm state, a fault can be recovered before the timeout elapses. This timeout should be long enough for hardware or software to recover from the fault. If the fault is not recovered before the timeout elapses, the FCU goes into Fault state.

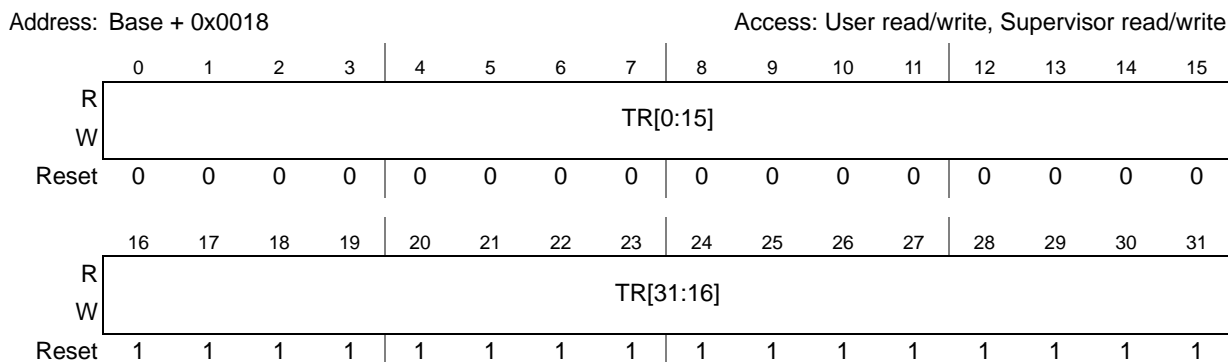


Figure 392. Timeout Register (FCU_TR)

Table 405. FCU_TR field descriptions

Field	Description
TR	FCU Timeout 00000: Timeout is one clock (16 MHz) cycle 00001: Timeout is one clock (16 MHz) cycle 00002: Timeout is two clock (16 MHz) cycles 00003: Timeout is three clock (16 MHz) cycles ... Default at reset is 0x0000_FFFF. Timeout is 65,535 clock cycles (about 4.1 ms at 16 MHz, high speed RC clock)

29.2.3.8 Timeout Enable Register (FCU_TER)

Once a specific fault is enabled, the user can select to move to Alarm or Fault state when a fault occurs. A timeout enable has no effect if the related fault enable flag is not set.

Address: Base + 0x001C

Access: User read/write, Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TESF	TESF	TESF	0	0	0	0	0	0	0	0	0	0	0
W			2	3	4											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF	TEHF
W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 393. Timeout Enable Register (FCU_TER)

Table 406. FCU_TER field descriptions

Field	Description
TESF2–TESF4	Timeout Enable for Software Recoverable Fault 0: FCU goes into Fault state on Software recoverable Fault[2:4] 1: FCU goes into Alarm state on Software recoverable Fault[2:4]
TEHF15–TEHF0	Timeout Enable for Hardware Recoverable Fault 0: FCU goes into Fault state on Hardware recoverable Fault[15:0] 1: FCU goes into Alarm state on Hardware recoverable Fault[15:0]

29.2.3.9 Module State Register (FCU_MSR)

The FCU_MSR indicates the current state of the FCU. Only one of these bits can be set at a time.

Address: Base + 0x0020

Access: User read-only, Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	S3	S2	S1	S0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 394. Module State Register (FCU_MSR)

Table 407. FCU_MSR field descriptions

Field	Description
S3	When this bit is set, the FCU is in the Fault state.
S2	When this bit is set, the FCU is in the Alarm state.
S1	When this bit is set, the FCU is in the Normal state.
S0	When this bit is set, the FCU is in the Init state.

29.2.3.10 Microcontroller State Register (FCU_MCSR)

The FCU_MCSR indicates the current state of the microcontroller in the MCSA field and the previous state (before state change) in the MCPS field. The contents of this register are copied into the Frozen MC State Register (FCU_FMCSR) when the FCU goes into Fault state.

Address: Base + 0x0024

Access: User read-only, Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MCPS[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MCAS[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 395. MC State Register (FCU_MCSR)
Table 408. FCU_MCSR field description

Field	Description
MCPS	MC Previous State 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111: Reserved

Table 408. FCU_MCSR field description

Field	Description
MCAS	MC Actual State 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111: Reserved

29.2.3.11 Frozen MC State Register (FCU_FMCSR)

The FCU_MCSR is copied into the FCU_FMCSR each time the Fault state is entered. The FCU_FMCSR bit description is the same as that of the FCU_MCSR.

Address: Base + 0x0028

Access: User read-only, Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FRMCPS[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FRMCAS[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 396. Frozen MC State Register (FCU_FMCSR)

Table 409. FCU_FMCSR field description

Field	Description
FRMCPS [3:0]	MC Previous State 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111: Reserved
FRMCAS [3:0]	MC Actual State 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111: Reserved

29.3 Functional description

The FCU module contains six functional blocks as shown in [Figure 397](#).

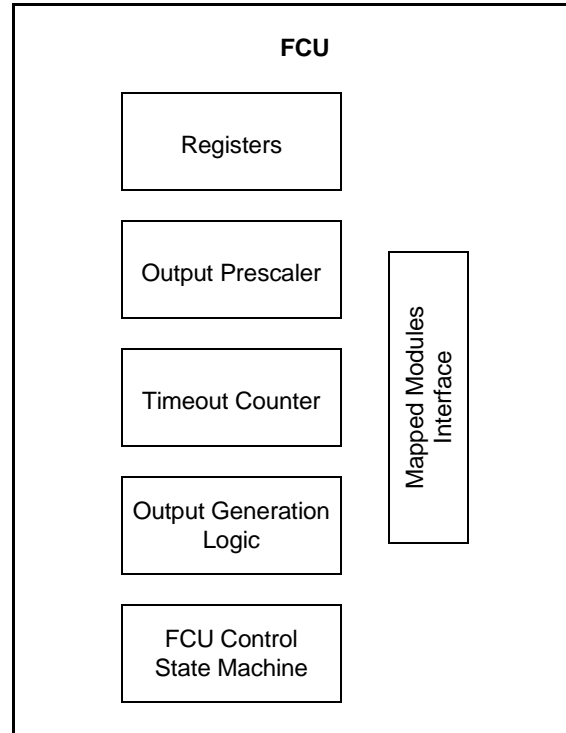


Figure 397. Functional block diagram

The register block implements all the FCU registers including input capture logic. The interface implements the read functionality and generated write and register enable signals. The timeout counter implements the counter to calculate timeout to switch from Alarm state to Fault state. The output prescaler module generates the prescaled clock to be used to generate output sequence. The FCU control implements the FCU state machine. The output generation logic generates two external output signals according to the output generation protocol.

29.3.1 State machine

The FCU provides four states: Init, Normal, Alarm, and Fault.

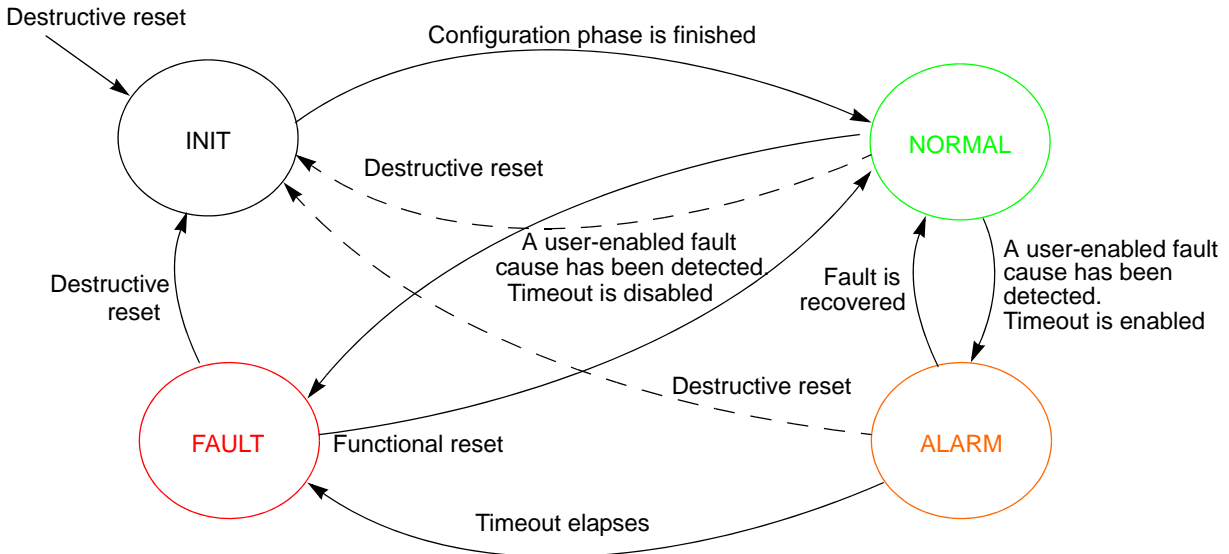


Figure 398. Finite state machine

The Init state is entered after any destructive reset assertion. Once the FCU goes into Fault state, both functional and destructive reset assertion lets the FCU move into Init state.

Once a configuration is locked (see [Section 29.2.3.1, “Module Configuration Register \(FCU_MCR\)”](#)), the FCU goes into Normal state. Then, when a fault occurs, the FCU can move into Alarm/Fault states depending on the Fault Enable Register (FCU_FER). The Fault Flag Register (FCU_FFR) stores any fault that has occurred, even if the FCU is not entering into the Alarm or the Fault state.

When the FCU is in Alarm state, a timer starts counting up to a fixed timeout (see [Section 29.2.3.7, “Timeout Register \(FCU_TR\)”](#)) before going into Fault state.

When the FCU goes into Fault state, the status of the FCU_MCR is copied into the FCU_FMCSR. Also, the FCU_FFR is copied into the FCU_FFFR.

The FCU can be tested by setting field TM[1:0] in the FCU_MCR during initialization. In this way, the FCU_FFGR can be accessed and faults can be emulated, since FCU behavior is the same whether real or fake faults occur. Optionally, the output pins can be disabled in Test mode (by setting the TM field to ‘01’). To exit from Test mode, the TM field must be set to ‘00’ or ‘11’.

After a functional reset, the FCU_FFR (not the Frozen Fault Flag Register (FCU_FFFR)) must be cleared and the FCU must return to Normal state.

29.3.2 Output generation protocol

The FCU provides two external output signals. The FCU supports different protocols for fault indication to the external device. Both external signals are used only in dual-rail protocol. In other protocols, the second output is the inverted version of the first output.

In all the protocols, depending on the polarity field (PS) in the MCR, the outputs can be normal polarity (if PS = 0) or inverted polarity (if PS = 1). The LSB of PS sets the polarity of the first signal; the MSB sets the polarity of the second signal.

29.3.2.1 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast to classical encoding, where each wire carries a single bit-value, dual-rail encoded circuits use two wires to carry each bit. [Table 410](#) summarizes the encoding.

Table 410. Dual-rail coding

Logical value	Dual-rail encoding (FCU[0], FCU[1])
non-faulty	(0,1)
non-faulty	(1,0)
faulty	(0,0)
faulty	(1,1)

As long as the FCU is in Normal or Alarm state, the output shows a “non-faulty” signal. FCU[0], FCU[1] toggles between (0,1) and (1,0) with a given frequency, set in the FOP field of the MCR. By default, this value is $f = 976 \text{ Hz @ } 64 \text{ MHz}$ (about 1 kHz, see [Equation 1](#)). The same frequency is used to show a faulty situation (FCU in Fault state).

In the Init state, output pins are set as high impedance by clearing the OBE bits for each pad in its respective PCR in the SIUL module.

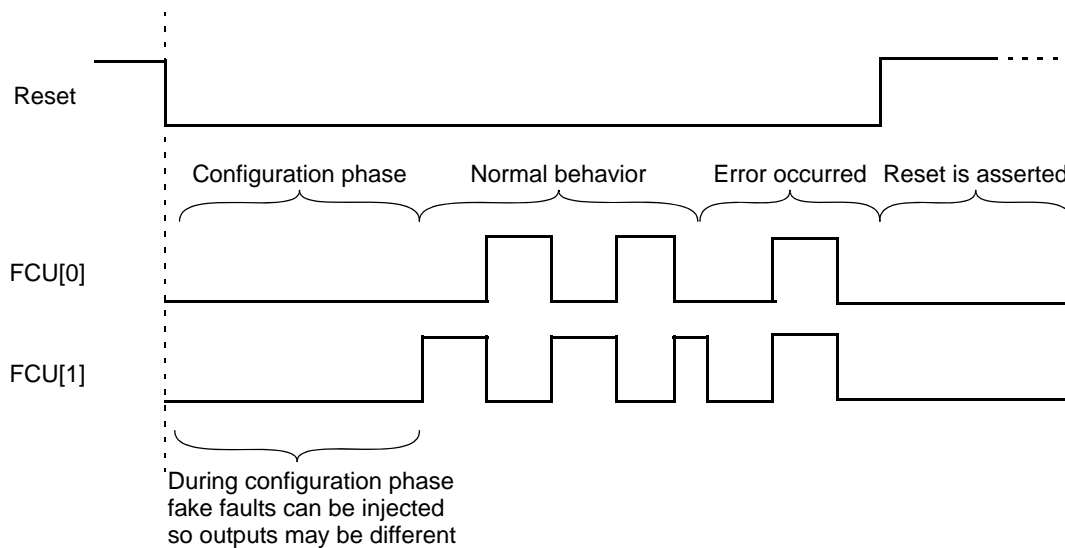


Figure 399. Dual rail coding example

29.3.2.2 Time switching protocol

FCU[0] is toggled between logic 0 and logic 1 with a defined frequency $f = 1 \text{ kHz @ } 64 \text{ MHz}$ (f is approximated, as shown by Equation 1) and duty cycle $d = 50\%$.

Eqn. 1

$$\text{EOUT freq} = \text{SYS_CLK} \div \{4096 \times [(\text{FOP} + 1) \times 2]\}$$

$$64 \text{ MHz} \div \{4096 \times [(7 + 1) \times 2]\} = 976.5 \text{ Hz}$$

Frequency can be varied by using the same prescaler as used for the dual-rail protocol (FOP field in FCU_MCR). This frequency modulation protocol is violated when the FCU goes into Fault state.

During initialization phase, FCU[0] is set high. When a fault is detected, FCU[0] is set low.

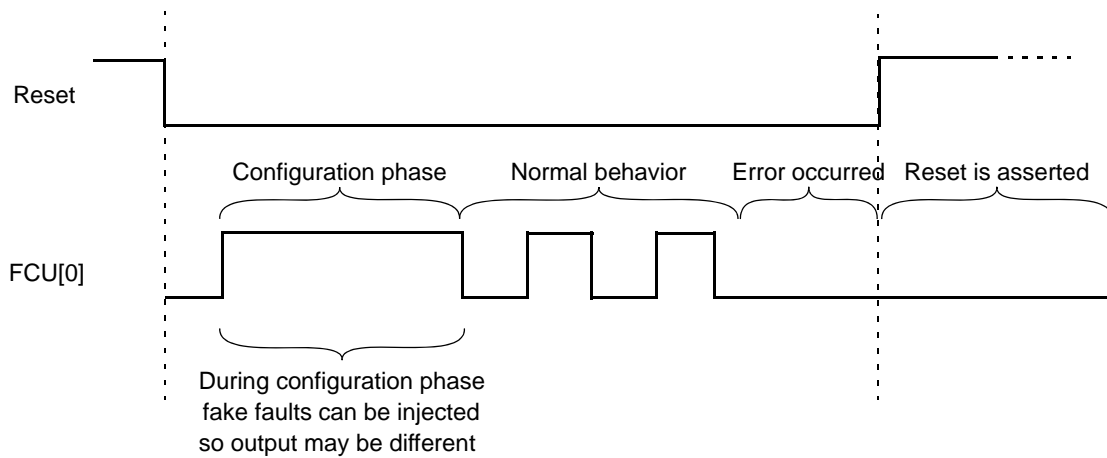


Figure 400. Time switching protocol example

29.3.2.3 Bi-Stable protocol

In this protocol during the Init and the Fault state, faulty state is indicated. In the Normal/Alarm state, non-faulty state is indicated. Table 411 shows bi-stable encoding for FCU[0].

Table 411. Bi-stable coding

Logical value	Bi-stable encoding
faulty	0
non-faulty	1

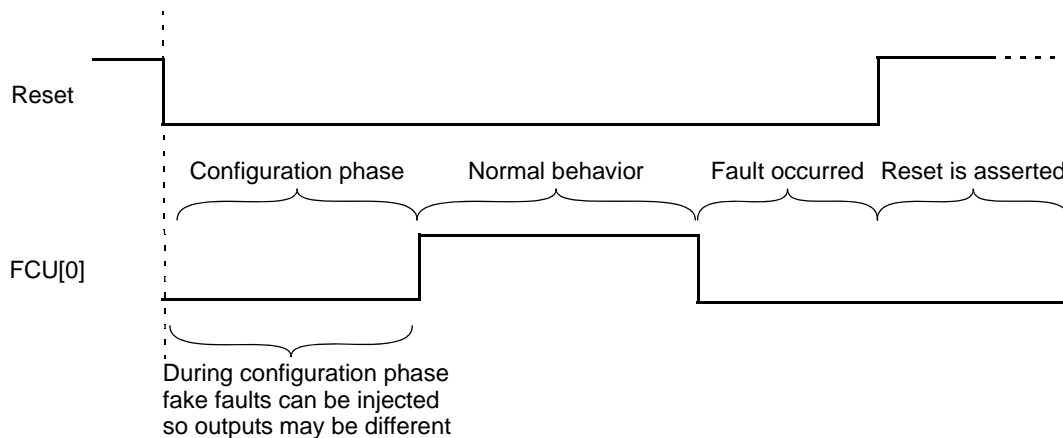


Figure 401. Bi-stable coding example

Chapter 30

Periodic Interrupt Timer (PIT_RTI)

30.1 Front Matter

30.1.1 Preface

The PIT block implements several timers which can be used for DMA triggering, general purpose interrupts and system wakeup.

30.1.1.1 Conventions

Table 412 is a list of conventions used in this document.

Table 412. Conventions

Terms	Description
ACTIVE_HIGH	Names for signals that are active high are shown in uppercase text without an overbar. Signals that are active high are referred to as asserted when they are logic 1 and negated when they are logic 0.
ACTIVE_LOW	A bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted when they are logic 0 and negated when they are logic 1.
0x0F	Hexadecimal numbers
0b0011	Binary numbers
x	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded 0bx001, the state of the first bit is a don't care.

30.1.1.2 Acronyms and Abbreviations

Table 413 contains acronyms and abbreviations used in this document.

Table 413. Acronyms and Abbreviations

Term	Definition
PIT	Period Interrupt Timer Module - A module which provides 10 general purpose timers and one RTI timer for system wakeup
CPU	Central Processor Unit
DMA	Direct Memory Access
ATD	Analog To Digital converter
CRG	Clock and Reset Generator module

30.1.1.3 Glossary

Table 414. Glossary

Term	Definition
Set	To set a bit or bits means to establish logic level one on the bit or bits.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

30.2 Introduction

The figure below shows the PIT block diagram.

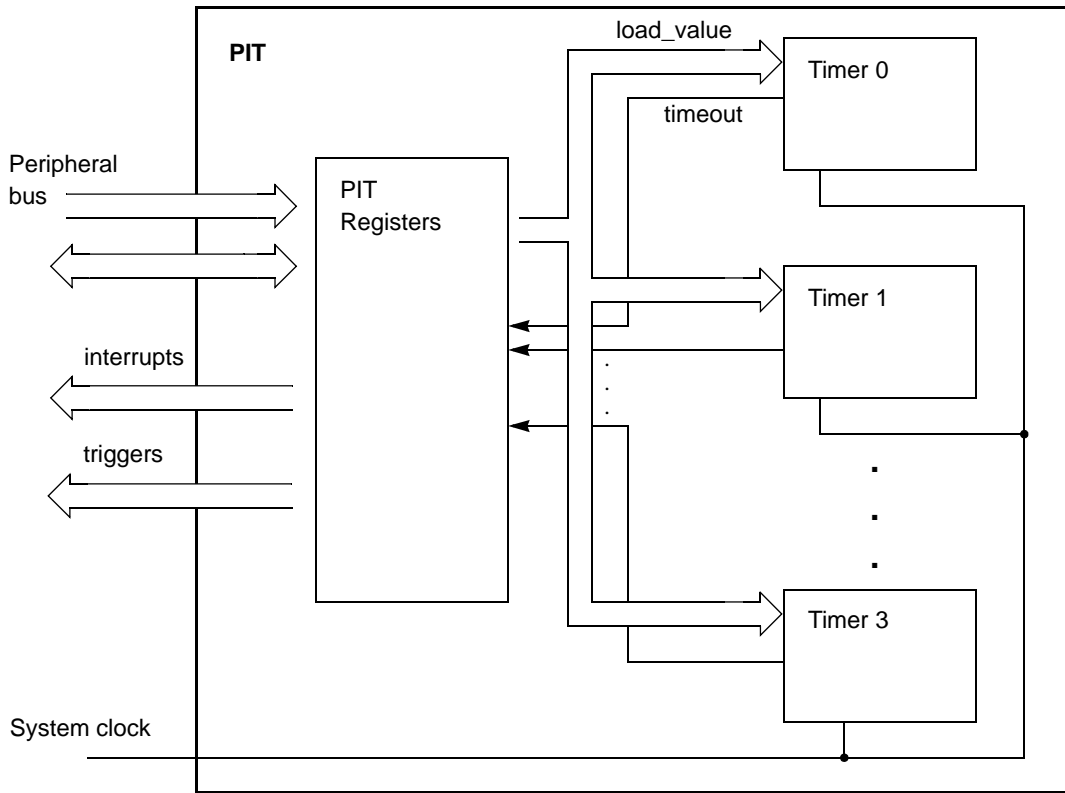


Figure 402. Block diagram of PIT_RTI

30.2.1 Overview

This specification describes the function of the Periodic Interrupt Timer block (PIT_RTI). The PIT is an array of timers that can be used to raise interrupts and trigger DMA channels. On some devices it provides a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup.

30.2.2 Features

The main features of this block are:

- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- Independent timeout periods for each timer

30.3 Signal Description

The PIT module has no external pins.

30.4 Memory Map and Register Description

This section provides a detailed description of all registers accessible in the PIT_RTI module.

30.4.1 Memory Map

Table 415 gives an overview on all PIT_RTI registers.

Table 415. PIT_RTI Memory Map

Address Offset	Use	Access
0x000	PIT Module Control Register	R/W
0x004 - 0x0FC	Reserved	R
Timer Channel 0		
0x0100	LDVAL0—Timer 0 Load Value Register	R/W
0x0104	CVAL0—Timer 0 Current Value Register	R
0x0108	TCTRL0—Timer 0 Control Register	R/W ¹
0x010C	TFLG0—Timer 0 Flag Register	R/W ¹
Timer Channel 1		
0x0110	LDVAL1—Timer 1 Load Value Register	R/W
0x0114	CVAL1—Timer 1 Current Value Register	R

Table 415. PIT_RTI Memory Map (continued)

Address Offset	Use	Access
0x0118	TCTRL1—Timer 1 Control Register	R/W ¹
0x011C	TFLG1—Timer 1 Flag Register	R/W ¹
Timer Channel 2		
0x0120	LDVAL2—Timer 2 Load Value Register	R/W
0x0124	CVAL2—Timer 2 Current Value Register	R
0x0128	TCTRL2—Timer 2 Control Register	R/W ¹
0x012C	TFLG2—Timer 2 Flag Register	R/W ¹
Timer Channel 3		
0x0130	LDVAL3—Timer 3 Load Value Register	R/W
0x0134	CVAL3—Timer 3 Current Value Register	R
0x0138	TCTRL3—Timer 3 Control Register	R/W ¹
0x013C	TFLG3—Timer 3 Flag Register	R/W ¹
0x140 - 0x1FC	Reserved	R

Table 416. Timer Channel n

Address Offset	Use	Access
channel + 0x00	Timer Load Value Register	R/W
channel + 0x04	Current Timer Value Register	R
channel + 0x08	Timer Control Register	R/W
channel + 0x0C	Timer Flag Register	R/W

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

NOTE

Reserved registers will read as 0, writes will have no effect.

30.4.2 Register Descriptions

This section describes in address order all the PIT_RTI registers and their individual bits.

30.4.2.1 PIT Module Control Register (PITMCR)

This register controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Offset 0x000

Access: Read/Write

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

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

Figure 403. PIT Module Control Registers (PITMCR)

Table 417. PITMCR Field Descriptions

Field	Description
MDIS	Module Disable. This is used to disable the module clock. This bit should be enabled before any other setup is done. 0 Clock for PIT Timers is enabled 1 Clock for PIT Timers is disabled (default)
FRZ	Freeze. Allows the timers to be stopped when the device enters debug mode. 0 = Timers continue to run in debug mode. 1 = Timers are stopped in debug mode.

30.4.2.2 Timer Load Value Register n (LDVALn)

These registers select the timeout period for the timer interrupts.

Offset channel_base + 0x00

Access: Read/Write

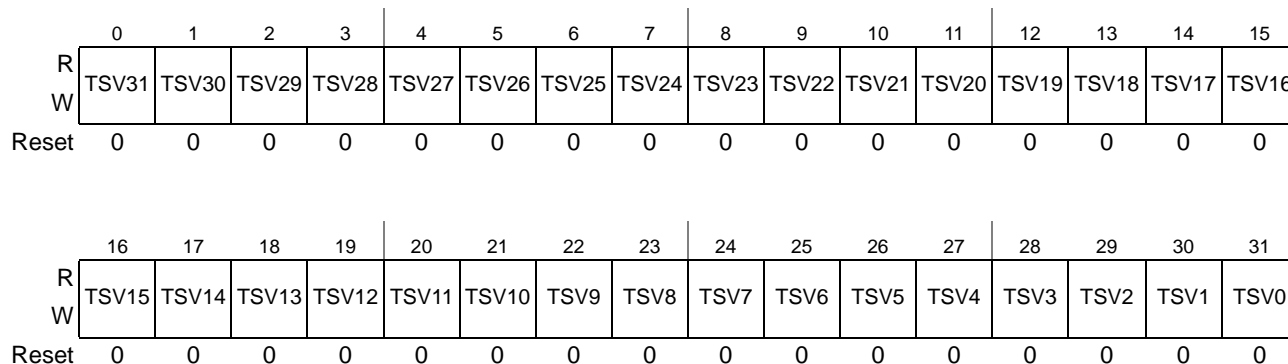


Figure 404. Timer Load Value Register n (LDVALn)

Table 418. LDVAL Field Descriptions

Field	Description
TSVn	Time Start Value Bits. These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again (see Figure 409).

30.4.2.3 Current Timer Value Register n (CVALn)

These registers indicate the current timer position.

Offset channel_base + 0x04

Access: Read

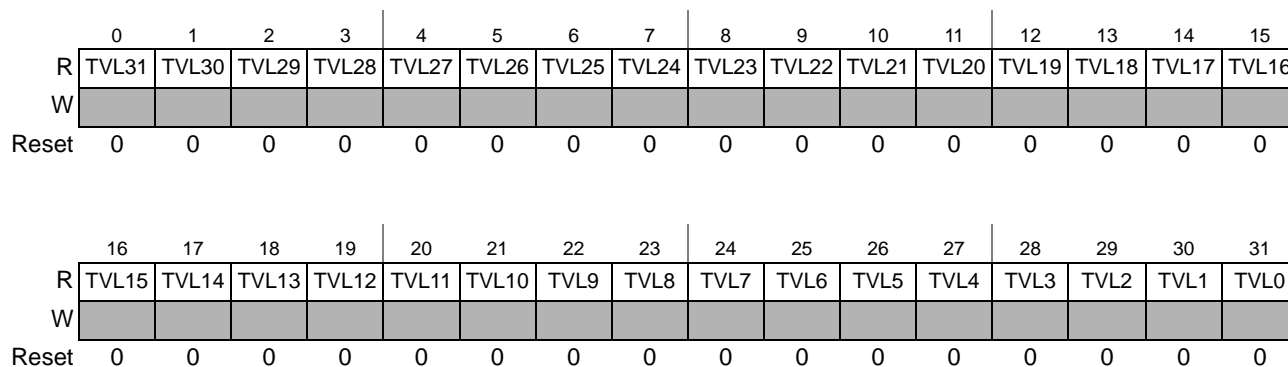


Figure 405. Current Timer Value Register n (CVALn)

Table 419. CVAL Field Descriptions

Field	Description
TVL _n	Current Timer Value. These bits represent the current timer value. Note that the timer uses a downcounter. NOTE: The timer values will be frozen in Debug mode if the FRZ bit is set in the PIT Module Control Register (see Figure 403)

30.4.2.4 Timer Control Register n (TCTRLn)

These registers contain the control bits for each timer.

Offset channel_base + 0x08

Access: Read/Write

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

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

Figure 406. Timer Control Register n (TCTRLn)
Table 420. TCTRL Field Descriptions

Field	Description
TIE	Timer Interrupt Enable Bit. 0 Interrupt requests from Timer x are disabled 1 Interrupt will be requested whenever TIF is set When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
TEN	Timer Enable Bit. 0 Timer will be disabled 1 Timer will be active

30.4.2.5 Timer Flag Register n (TFLGn)

These registers hold the PIT interrupt flags.

Offset channel_base + 0x0C

Access: Read/Write

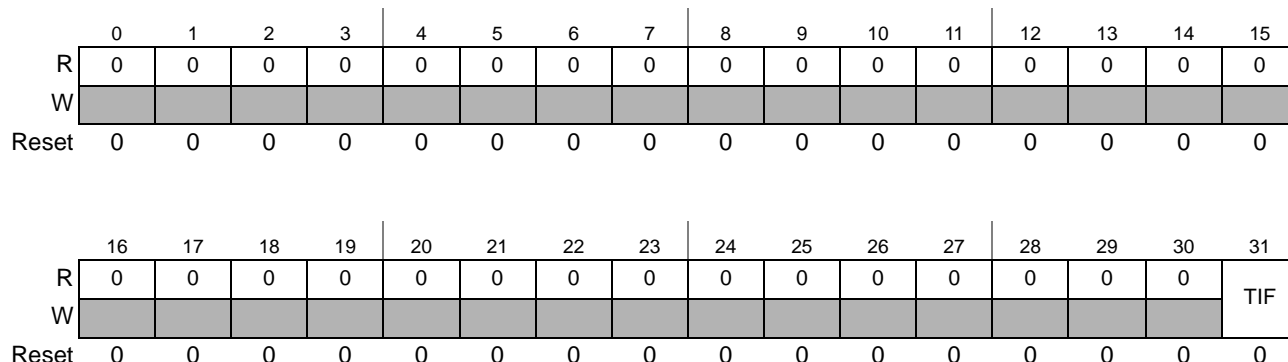


Figure 407. Timer Flag Register n (TFLGn)

Table 421. TFLG Field Descriptions

Field	Description
TIF	Time Interrupt Flag. TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with a 1. Writing a 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Time-out has not yet occurred 1 Time-out has occurred

30.5 Functional Description

30.5.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts, each interrupt will be available on a separate interrupt line.

30.5.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse, and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see [Figure 408](#)).

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see [Figure 409](#)).

It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (see [Figure 410](#)).

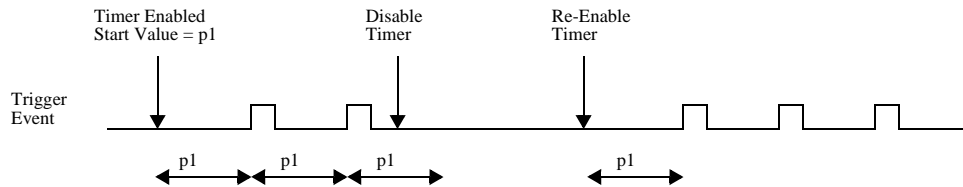


Figure 408. Stopping and Starting a Timer

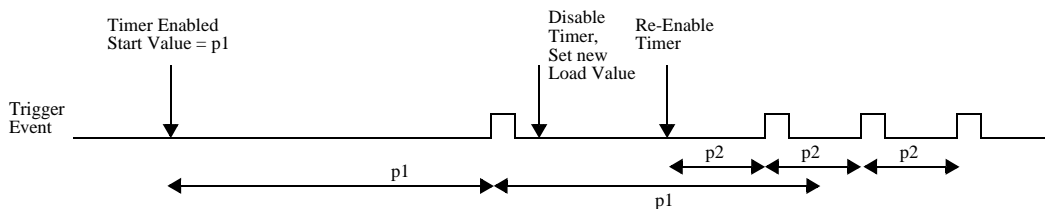


Figure 409. Modifying Running Timer Period

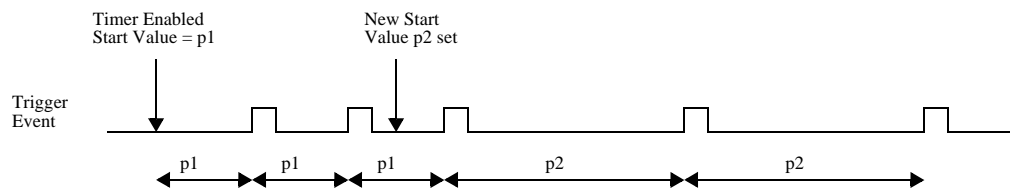


Figure 410. Dynamically Setting a New Load Value

30.5.1.2 Debug Mode

In Debug Mode the timers will be frozen - this is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (e.g. the timer values) and then continue the operation.

30.5.2 Interrupts

All of the timers support interrupt generation. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be disabled by setting the TIE bits to zero. The timer interrupt flags (TIF) are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to that TIF bit.

30.6 Initialization and Application Information

30.6.1 Example Configuration

In the example configuration:

- the PIT clock has a frequency of 50 MHz
- timer 1 shall create an interrupt every 5.12 ms
- timer 3 shall create a trigger event every 30 ms

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the PITCTRL register.

The 50 MHz clock frequency equates to a clock period of 20 ns . Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles. The value for the LDVAL register trigger would be calculated as (period / clock period) -1.

This means that LDVAL1 with 0003E7FF hex and LDVAL3 with 0016E35F hex.

The interrupt for Timer 1 is enabled by setting TIE in the TCTRL1 register. The timer is started by writing a 1 to bit TEN in the TCTRL1 register.

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a 1 to bit TEN in the TCTRL3 register, bit TIE stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_CTRL = 0x00;

// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
PIT_TCTRL3 = TEN; // start timer 3
```


Chapter 31

Software Watchdog Timer (SWT)

31.1 Introduction

31.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out, a reset is always generated on a second consecutive time-out.

31.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of fixed or keyed servicing
- Master access protection
- Hard and soft configuration lock bits

31.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_MCR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_MCR. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

31.2 External signal description

The SWT module does not have any external interface signals.

31.3 Memory map and register definition

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_MCR is set then the SWT generates a system reset on an invalid access otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_MCR are set then the SWT_MCR, SWT_TO, SWT_WN, SWT_SK registers are read-only.

31.3.1 Memory map

The SWT memory map is shown in [Table 422](#). The reset values of SWT_MCR, SWT_TO and SWT_WN are device specific. These values are determined by SWT inputs.

Table 422. SWT memory map

Offset from SWT Base address (0xFFF3_8000)	Register name	Register description	Size (bits)	Access	Location
0x0000	SWT_MCR	SWT Module Control Register	32	R/W	on page 758
0x0004	SWT_IR	SWT Interrupt Register	32	R/W	on page 760
0x0008	SWT_TO	SWT Time-out Register	32	R/W	on page 760
0x000C	SWT_WN	SWT Window Register	32	R/W	on page 761
0x0010	SWT_SR	SWT Service Register	32	R/W	on page 761
0x0014	SWT_CO	SWT Counter Output Register	32	R	on page 762
0x0018	SWT_SK	SWT Service Key Register	32	R/W	on page 762
0x001C – 0x3FFF	Reserved		—	—	—

31.3.2 Register descriptions

The following sections detail the individual registers within the SWT programming model.

31.3.2.1 SWT Module Control Register (SWT_MCR)

The SWT_MCR contains fields for configuring and controlling the SWT. The reset value of this register is device specific. Some devices can be configured to automatically clear the SWT_MCR[WEN] bit during the boot process. This register is read-only if either the SWT_MCR[HLK] or SWT_MCR[SLK] bits are set.

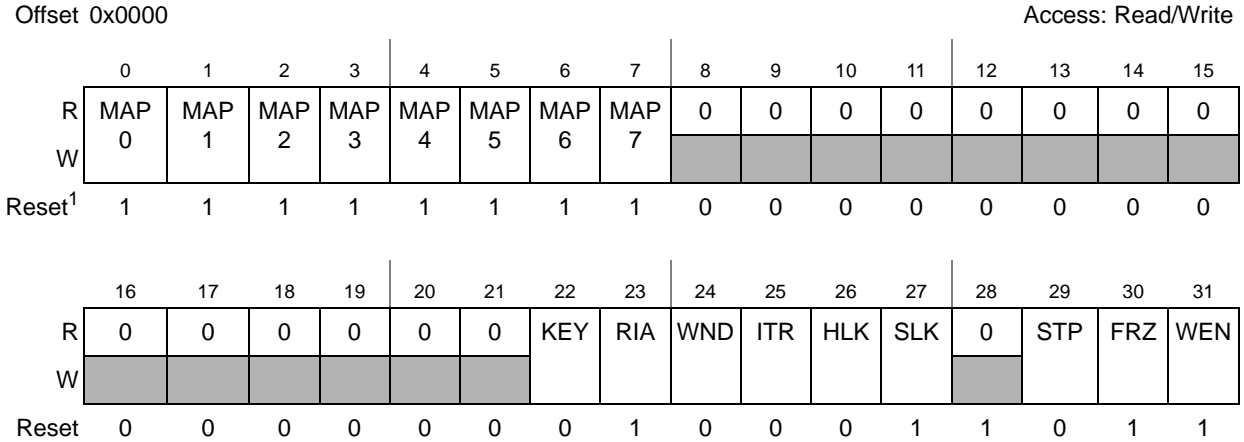


Figure 411. SWT Control Register (SWT_MCR)

Table 423. SWT_MCR field description

Field	Description
MAPn	Master Access Protection for Master n The platform bus master assignments are device specific. 0 = Access for the master is not enabled 1 = Access for the master is enabled
KEY	Keyed Service Mode 0 = Fixed Service Sequence, the fixed sequence 0xA602, 0xB480 is used to service the watchdog 1 = Keyed Service Mode, two pseudorandom key values are used to service the watchdog
RIA	Reset on Invalid Access 0 = Invalid access to the SWT generates a bus error 1 = Invalid access to the SWT causes a system reset if WEN = 1
WND	Window Mode 0 = Regular mode, service sequence can be done at any time 1 = Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
ITR	Interrupt Then Reset 0 = Generate a reset on a time-out 1 = Generate an interrupt on an initial time-out, reset on a second consecutive time-out
HLK	Hard Lock This bit is only cleared at reset. 0 = SWT_MCR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK = 0 1 = SWT_MCR, SWT_TO, SWT_WN and SWT_SK are read-only registers
SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 = SWT_MCR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK = 0 1 = SWT_MCR, SWT_TO, SWT_WN and SWT_SK are read-only registers
STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode 0 = SWT counter continues to run in stop mode 1 = SWT counter is stopped in stop mode

Table 423. SWT_MCR field description (continued)

Field	Description
FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode 0 = SWT counter continues to run in debug mode 1 = SWT counter is stopped in debug mode
WEN	Watchdog Enabled 0 = SWT is disabled 1 = SWT is enabled

31.3.2.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

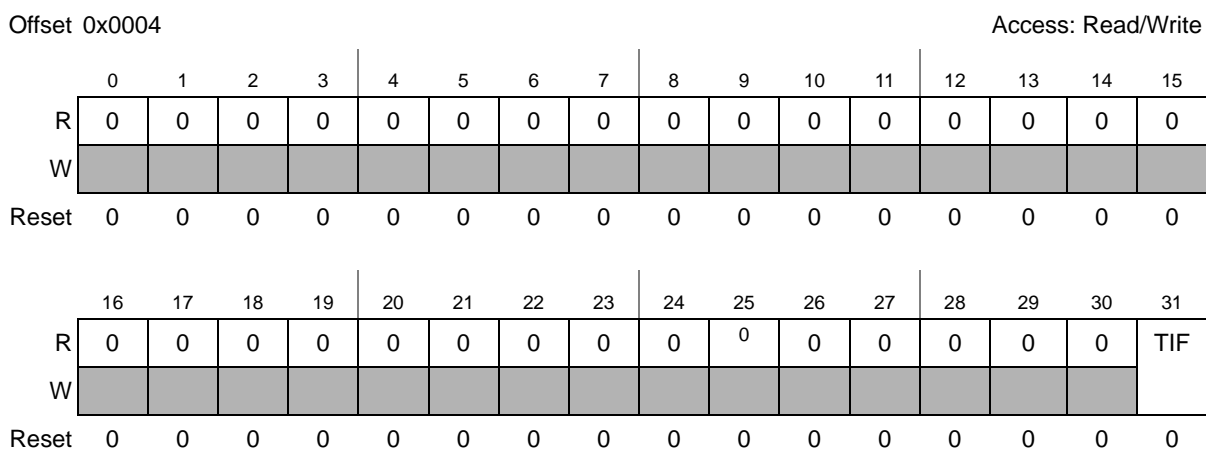


Figure 412. SWT Interrupt Register (SWT_IR)

Table 424. SWT_IR field description

Field	Description
TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a '1' to this bit. Writing a '0' has no effect. 0 = No interrupt request 1 = Interrupt request due to an initial time-out

31.3.2.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. The reset value for this register is device specific. This register is read-only if either the SWT_MCR[HLK] or SWT_MCR[SLK] bits are set.

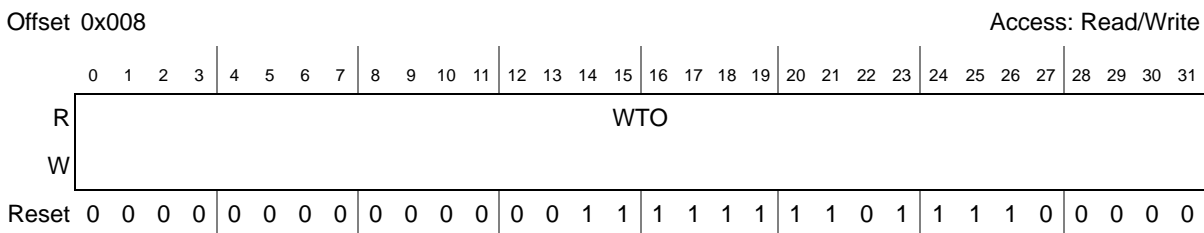


Figure 413. SWT Time-Out Register (SWT_TO)

Table 425. SWT_TO Register field description

Field	Description
WTO	Watchdog time-out period in clock cycles An internal 32-bit down counter is loaded with this value or 0x100 which ever is greater when the service sequence is written or when the SWT is enabled.

31.3.2.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_MCR[HLK] or SWT_MCR[SLK] bits are set.

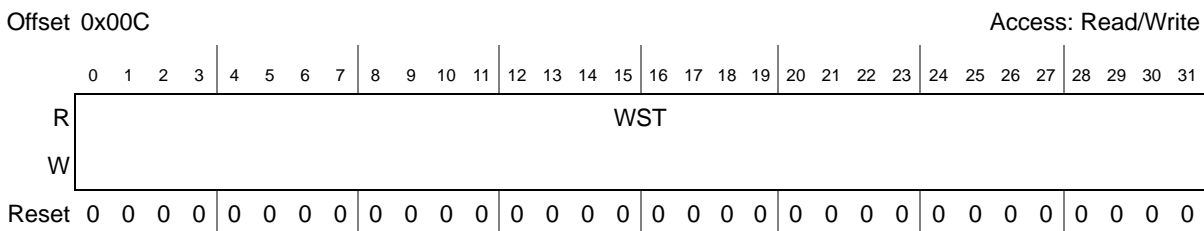


Figure 414. SWT Window Register (SWT_WN)

Table 426. SWT_WN Register field description

Field	Description
WST	Window start value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

31.3.2.5 SWT Service Register (SWT_SR)

The SWT Time-Out (SWT_SR) service register is the target for service operation writes used to reset the watchdog timer.

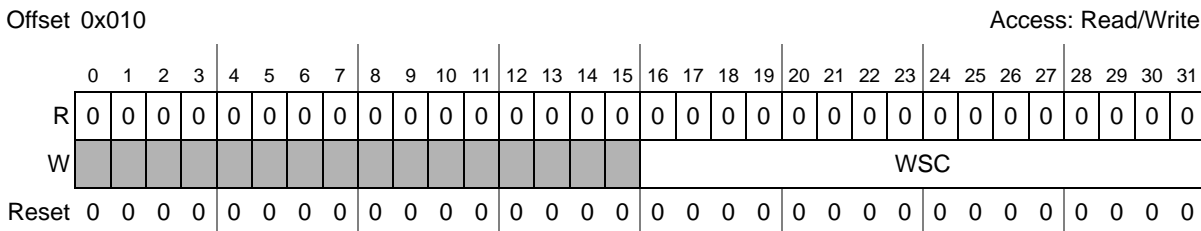


Figure 415. SWT Service Register (SWT_SR)

Table 427. SWT_SR field description

Field	Description
WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_MCR[SLK]). If the SWT_MCR[KEY] bit is set, two pseudorandom key values are written to service the watchdog, see Section 31.4, "Functional description" , for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_MCR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field.

31.3.2.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

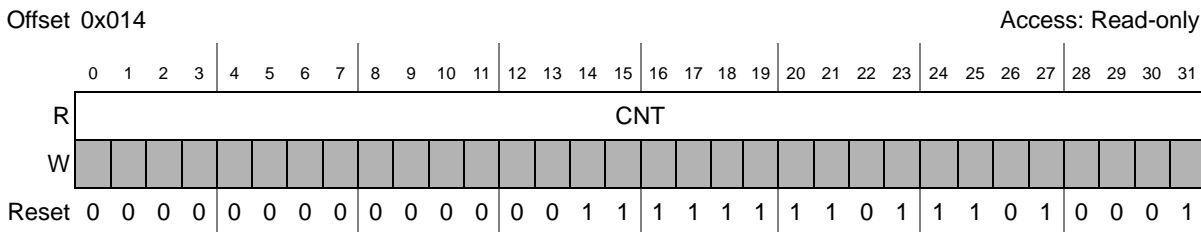


Figure 416. SWT Counter Output Register (SWT_CO)

Table 428. SWT_CO Register field description

Field	Description
CNT	Watchdog Count When the watchdog is disabled (SWT_MCR[WEN] = 0) this field shows the value of the internal down counter. When the watchdog is enabled the value of this field is 0x0000_0000. Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

31.3.2.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_MCR[HLK] or SWT_MCR[SLK] bits are set.

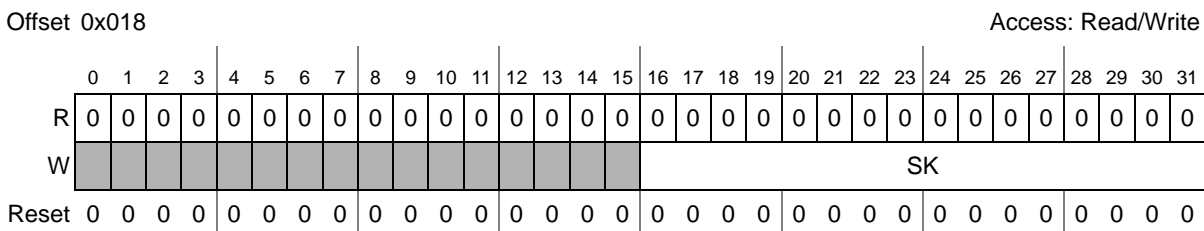


Figure 417. SWT Service Register (SWT_SR)

Table 429. SWT_SR field description

Field	Description
SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_MCR[KEY] is set, the next key value to be written to the SWT_SR is $(17*SK+3) \bmod 2^{16}$.

31.4 Functional description

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes a control register (SWT_MCR), an interrupt register (SWT_IR), a time-out register (SWT_TO), a window register (SWT_WN), a service register (SWT_SR), a counter output register (SWT_CO) and a service key register (SWT_SK).

The SWT_MCR includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_MCR[WEN] bit. The reset value of the SWT_MCR[WEN] bit is dependent upon the SWT field in the Reset Configuration Half Word. If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released.

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100 in which case the time-out period is set to 0x100. This time-out period is loaded into an internal 32-bit down counter when the SWT is enabled and each time a valid service operation is performed. The IRC clock is used to drive the down counter.

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked the SWT_MCR, SWT_TO, SWT_WN and SWT_SK registers are read-only. The hard lock is enabled by setting the SWT_MCR[HCLK] bit which can only be cleared by a reset. The soft lock is enabled by setting the SWT_MCR[SLK] bit and is cleared by writing the unlock sequence to the service register. The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_MCR[WEN] bit to be set.

When enabled, the SWT requires periodic execution of a servicing operation which consists of writing two values to the SWT_SR. Writing the proper sequence of values loads the internal down counter with the time-out period. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes. If the SWT_MCR[KEY] bit is zero, the fixed sequence 0xA602, 0xB480 is written to the SWT_SR[WSC] field to service the watchdog. If the SWT_MCR[KEY] bit is set, then two pseudorandom keys are written to the SWT_SR[WSC] field to service the watchdog. The key values

are determined by the pseudorandom key generator defined in [Figure 418](#). This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100 then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 * SK_n + 3) \text{ mod } 2^{16}$$

Figure 418. Pseudorandom key generator

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

If window mode is enabled (SWT_MCR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_MCR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000 then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

The interrupt then reset bit (SWT_MCR[ITR]) controls the action taken when a time-out occurs. If the SWT_MCR[ITR] bit is not set, a reset is generated immediately on a time-out. If the SWT_MCR[ITR] bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR[TIF]). The interrupt request is cleared by writing a '1' to the SWT_IR[TIF] bit.

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_MCR[WEN] cleared) and the value of the SWT_CO read to determine if the internal down counter is working properly.

Chapter 32

System Timer Module (STM)

32.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

32.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

32.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

32.4 External signal description

The STM does not have any external interface signals.

32.5 Memory map and registers description

The STM programming model has fourteen 32-bit registers. The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination. STM registers are accessible only when the core is in supervisor mode.

32.5.1 Memory map

The STM memory map is shown in [Table 430](#). The addresses presented here are the offsets relative to the base address 0xFFF3_C000.

Table 430. STM memory map

Offset from STM_BASE (0xFFF3_C000)	Register	Access	Reset value	Location
0x0000	STM_CR—STM Control Register	R/W	0x0000_0000	on page 766
0x0004	STM_CNT—STM Counter Value	R/W	0x0000_0000	on page 767
0x0008–0x000F	Reserved			
0x0010	STM_CCR0—STM Channel 0 Control Register	R/W	0x0000_0000	on page 768
0x0014	STM_CIR0—STM Channel 0 Interrupt Register	R/W	0x0000_0000	on page 768
0x0018	STM_CMP0—STM Channel 0 Compare Register	R/W	0x0000_0000	on page 769
0x001C	Reserved			
0x0020	STM_CCR1—STM Channel 1 Control Register	R/W	0x0000_0000	on page 768
0x0024	STM_CIR1—STM Channel 1 Interrupt Register	R/W	0x0000_0000	on page 768
0x0028	STM_CMP1—STM Channel 1 Compare Register	R/W	0x0000_0000	on page 769
0x002C	Reserved			
0x0030	STM_CCR2—STM Channel 2 Control Register	R/W	0x0000_0000	on page 768
0x0034	STM_CIR2—STM Channel 2 Interrupt Register	R/W	0x0000_0000	on page 768
0x0038	STM_CMP2—STM Channel 2 Compare Register	R/W	0x0000_0000	on page 769
0x003C	Reserved			
0x0040	STM_CCR3—STM Channel 3 Control Register	R/W	0x0000_0000	on page 768
0x0044	STM_CIR3—STM Channel 3 Interrupt Register	R/W	0x0000_0000	on page 768
0x0048	STM_CMP3—STM Channel 3 Compare Register	R/W	0x0000_0000	on page 769
0x004C–0x3FFF	Reserved			

32.5.2 Registers description

The following sections detail the individual registers within the STM programming model.

32.5.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Address: Base + 0x0000

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS[7:0]								0	0	0	0	0	0	FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 419. STM Control Register (STM_CR)

Table 431. STM_CR field descriptions

Field	Description
CPS[7:0]	Counter Prescaler Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1. 0x01 Divide system clock by 2. ... 0xFF Divide system clock by 256.
FRZ	Freeze Allows the timer counter to be stopped when the device enters debug mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
TEN	Timer Counter Enabled 0 Counter is disabled. 1 Counter is enabled.

32.5.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Address: Base + 0x0004

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 420. STM Count Register (STM_CNT)

Table 432. STM_CNT field descriptions

Field	Description
CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

32.5.2.3 STM Channel Control Register (STM_CCRn)

The STM Channel Control Register (STM_CCRn) enables and services channel *n* of the timer.

Address: Base + 0x0010 (STM_CCR0)
 Base + 0x0020 (STM_CCR1)
 Base + 0x0030 (STM_CCR2)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 421. STM Channel Control Register (STM_CCRn)

Table 433. STM_CCRn field descriptions

Field	Description
CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

32.5.2.4 STM Channel Interrupt Register (STM_CIRn)

The STM Channel Interrupt Register (STM_CIRn) enables and services channel *n* of the timer.

Address: Base + 0x0014 (STM_CIR0)
 Base + 0x0024 (STM_CIR1)
 Base + 0x0034 (STM_CIR2)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CIF[0]
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 422. STM Channel Interrupt Register (STM_CIRn)

System Timer Module (STM)

The channel is enabled by setting the `STM_CCR n [CEN]` bit. When enabled, the channel will set the `STM_CIR[CIF]` bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the `STM_CIR n [CIF]` bit. A write of 0 to the `STM_CIR n [CIF]` bit has no effect.

Chapter 33

Cyclic Redundancy Check (CRC)

33.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, thus off-loading the CPU. The MPC5604E CRC supports two contexts. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait states insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of two hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/half-word/word) formats.

33.1.1 Glossary

- CRC: cyclic redundancy check
- CPU: central processing unit
- DMA: direct memory access
- CCITT: ITU-T (for Telecommunication Standardization Sector of the International Telecommunications Union)
- SW: software
- WS: wait state
- SPI: serial peripheral interface

33.2 Main features

- 2 contexts for the concurrent CRC computation
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- 2 hard-wired polynomials (CRC-8 VDA CAN, CRC-16-CCITT and CRC-32 ethernet)
- Support for byte/half-word/word width of the input data stream

33.2.1 Standard features

- IPS bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT

- CRC-32 ethernet

33.3 Block diagram

Figure 424 shows the top level diagram of the CRC unit.

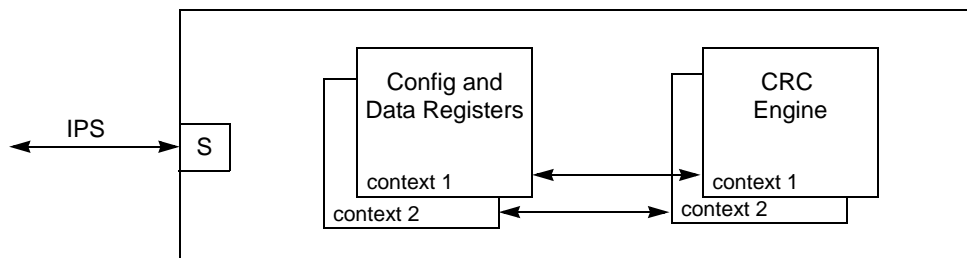


Figure 424. CRC top level diagram

33.3.1 IPS bus interface

The IPS bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32-bit) data write/read operations to any registers
- Low and high half-words (16-bit, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8-bit, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral.
- Any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.

The registers of the CRC module are accessible (read/write) in each access mode: user, supervisor, or test.

NOTE

For 8-bit CRC mode, use 8-bit access to registers to get the correct result data.

In terms of bus performance of the operations, following the summary:

- 0 WS (single bus cycle) for each write/read operations to the CRC_CFG and CRC_INP registers
- 0 WS (single bus cycle) for each write operation to the CRC_CSTAT register

- Double WS (3 bus cycles) for each read operation to the CRC_CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP or CRC_CFG registers belonging to the same context. In all the other cases no WS are inserted.

33.4 Functional description

The CRC module supports the CRC computation for each context. Each context has a own complete set of registers including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of byte, half-words or words. The input data sequence is provided, eventually mixing the data formats (byte, half-word, word), writing to the input data register (CRC_INP).

The data stream is generally executed by N concurrent DMA data transfers (mem2mem) where N is less or equal to the number of contexts.

Two standard generator polynomials are given in Equation 1 Equation 2 and Equation 3 for the CRC computation of each context.

CRC-8 VDA CAN) Eqn. 1

$$X^8 + X^4 + X^3 + X^2 + 1$$

CRC-16-CCITT (x25 protocol) Eqn. 2

$$X^{16} + X^{12} + X^5 + 1$$

CRC-32 (ethernet protocol) Eqn. 3

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

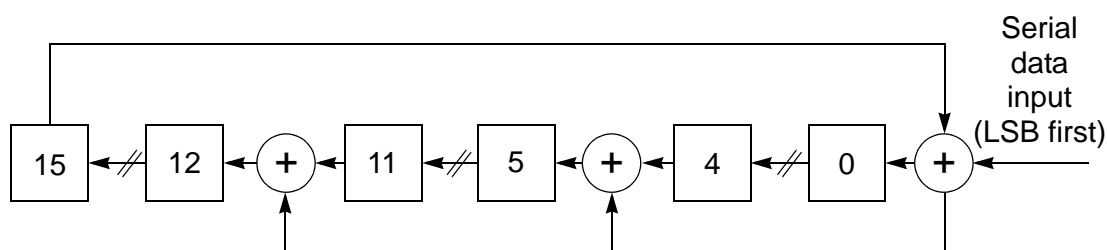


Figure 425. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed initializing the CRC_CSTAT register. The concept scheme (serial data loading) of the CRC engine is given in Figure 425 for the CRC-CCITT. The design implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to de-couple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

In case of usage of the CRC signature for encapsulation in the data frame of a communication protocol (e.g., SPI, ..) a bit swap (MSB → LSB, LSB → MSB) and/or bit inversion of the final CRC signature can be applied (CRC_OUTP register) before to transmit the CRC.

The usage of the CRC is summarized in the flow-chart given in [Figure 426](#).

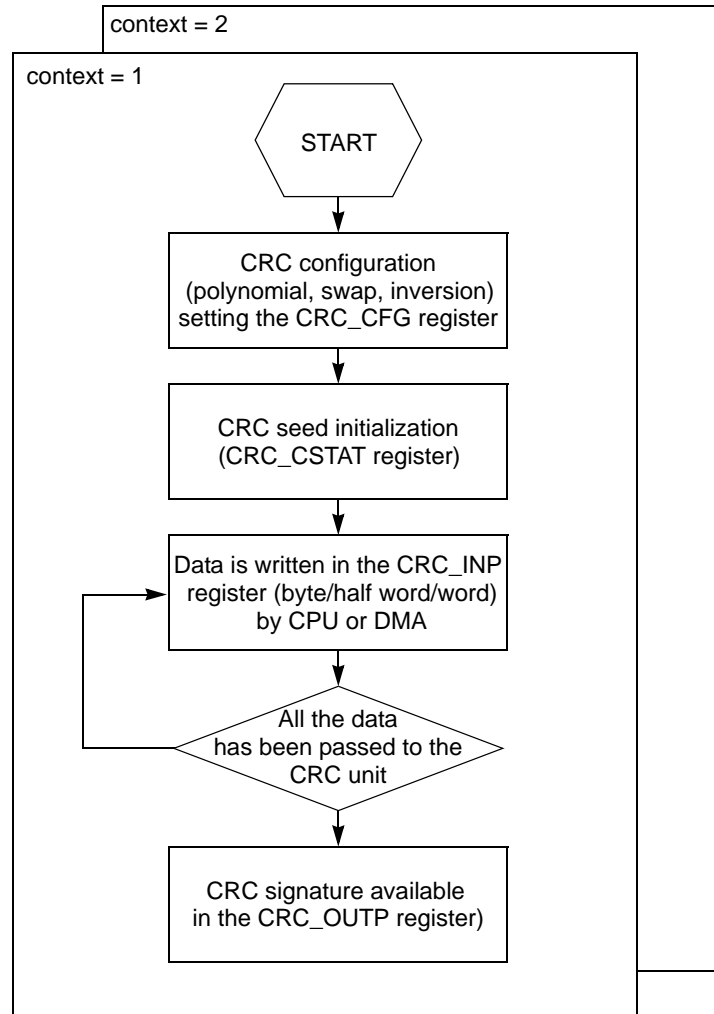


Figure 426. CRC computation flow

33.5 Memory map and registers description

[Table 436](#) shows the CRC memory map.

Table 436. CRC memory map

Offset from CRC_BASE (0xFFE6_8000)	Register	Location
0x0000	CRC_CFG—CRC Configuration Register, Context 1	on page 775
0x0004	CRC_INP—CRC Input Register, Context 1	on page 776
0x0008	CRC_CSTAT—CRC Current Status Register, Context 1	on page 777
0x000C	CRC_OUTP—CRC Output Register, Context 1	on page 777
0x0010	CRC_CFG—CRC Configuration Register, Context 2	on page 775
0x0014	CRC_INP—CRC Input Register, Context 2	on page 776
0x0018	CRC_CSTAT—CRC Current Status Register, Context 2	on page 777
0x001C	CRC_OUTP—CRC Output Register, Context 2	on page 777
0x0020–0x3FFF	Reserved	

33.5.1 CRC Configuration Register (CRC_CFG)

Address: Context 1: Base + 0x0000
Context 2: Base + 0x0010

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset ¹	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	POLY	SWAP	INV	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 427. CRC Configuration Register (CRC_CFG)

¹ The reset value of CRC_CFG(N), where N is the context number is as follows:

00000000h N = 0

00000004h N = 1

Table 437. CRC_CFG field descriptions

Field	Description
0:27	Reserved These are reserved bits. These bits are always read as 0 and must always be written with 0.
28:29	POLYG: <i>Polynomial selection</i> 00: CRC-CCITT polynomial. 01: CRC-32 polynomial. 10: CRC-8 polynomial. 11: CRC-8 polynomial. This bit can be read and written by the software. This bit can be written only during the configuration phase.
30	SWAP: <i>SWAP selection</i> 0: No swap selection applied on the CRC_OUTP content 1: Swap selection (MSB -> LSB, LSB -> MSB) applied on the CRC_OUTP content. In case of CRC-CCITT polynomial the swap operation is applied on the 16 LSB bits. This bit can be read and written by the software. This bit can be written only during the configuration phase.
31	INV: <i>INV selection</i> 0: No inversion selection applied on the CRC_OUTP content 1: Inversion selection (bit x bit) applied on the CRC_OUTP content. In case of CRC-CCITT polynomial the inversion operation is applied on the 16 LSB bits. This bit can be read and written by the software. This bit can be written only during the configuration phase.

33.5.2 CRC Input Register (CRC_INP)

Address: Context 1: Base + 0x0004
Context 2: Base + 0x0014

Access: User read/write

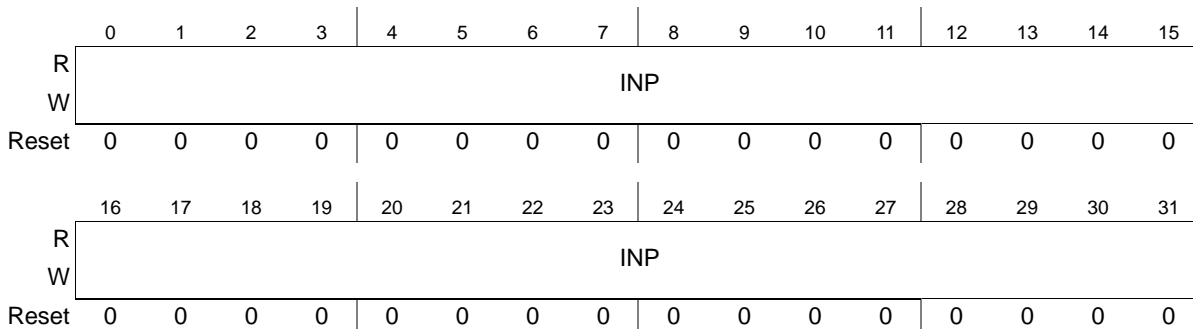


Figure 428. CRC Input Register (CRC_INP)

Table 438. CRC_INP field descriptions

Field	Description
0:31	INP: <i>Input data for the CRC computation</i> The INP register can be written at byte, half-word (high and low) or word in any sequence. In case of half-word write operation, the bytes must be contiguous. This register can be read and written by the software.

33.5.3 CRC Current Status Register (CRC_CSTAT)

Address: Context 1: Base + 0x0008
Context 2: Base + 0x0018

Access: User read/write

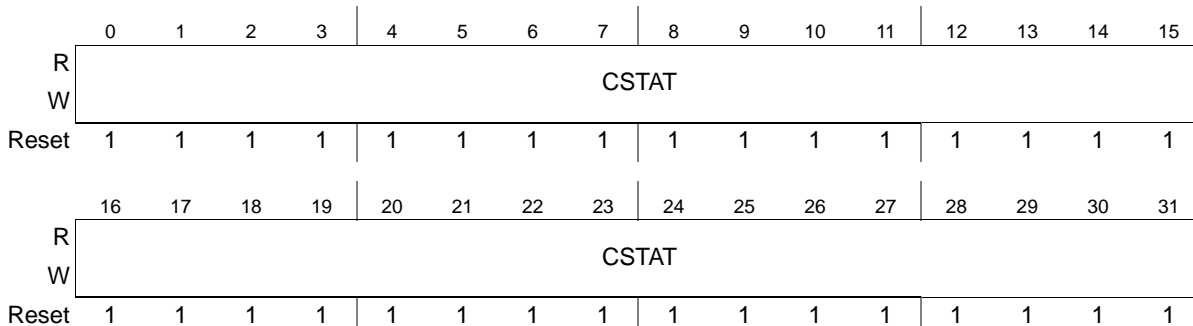


Figure 429. CRC Current Status Register (CRC_CSTAT)

Table 439. CRC_CSTAT field descriptions

Field	Description
0:31	<p>CSTAT: Status of the CRC signature</p> <p>The CSTAT register includes the current status of the CRC signature. No bit swap and inversion are applied to this register.</p> <p>In case of CRC-CCITT polynomial only the 16 LSB bits are significant. The 16 MSB bits are tied at 0b during the computation.</p> <p>The CSTAT register can be written at byte, half-word or word.</p> <p>This register can be read and written by the software.</p> <p>This register can be written only during the configuration phase.</p>

33.5.4 CRC Output Register (CRC_OUTP)

Address: Context 1: Base + 0x000C
Context 2: Base + 0x001C

Access: User read/write

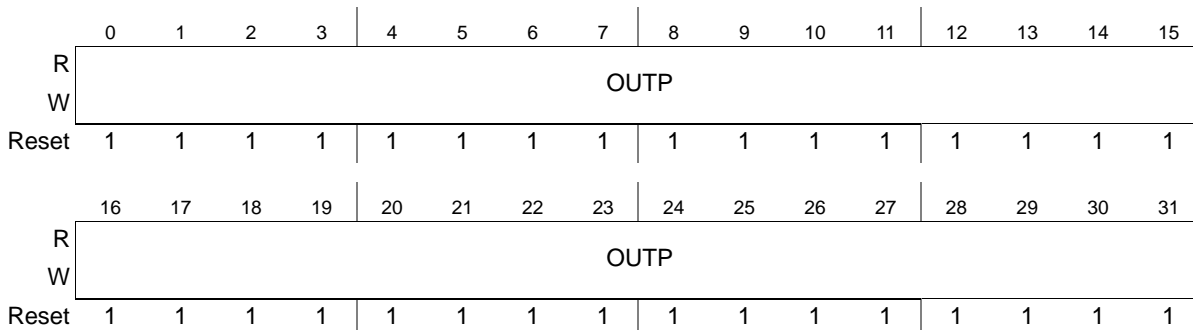


Figure 430. CRC Output Register (CRC_OUTP)

Table 440. CRC_OUTP field descriptions

Field	Description
0:31	<p>OUTP: <i>Final CRC signature</i></p> <p>The OUTP register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted.</p> <p>In case of CRC-CCITT polynomial only the 16 LSB bits are significant. The 16 MSB bits are tied at 0b during the computation.</p> <p>This register can be read by the software.</p>

33.6 Use cases and limitations

Two main use cases shall be considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outgoing frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral.

The signature of the configuration registers is computed in a correct way only if these registers do not contain any status bit. Assuming that the DMA engine has N channels (greater or equal to the number of contexts) configurable for the following type of data transfer: mem2mem, periph2mem, mem2periph, the following sequence, as given in [Figure 431](#), shall be applied to manage the transmission data flow:

- DMA/CRC module configuration (context x, channel x) by CPU
- Payload transfer from the MEM to the CRC module (CRC_INP register) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
- CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
- Data block (payload + CRC) transfer from the MEM to the PERIPH module (e.g., SPI Tx fifo) (phase 3) by DMA (mem2periph data transfer, channel x)

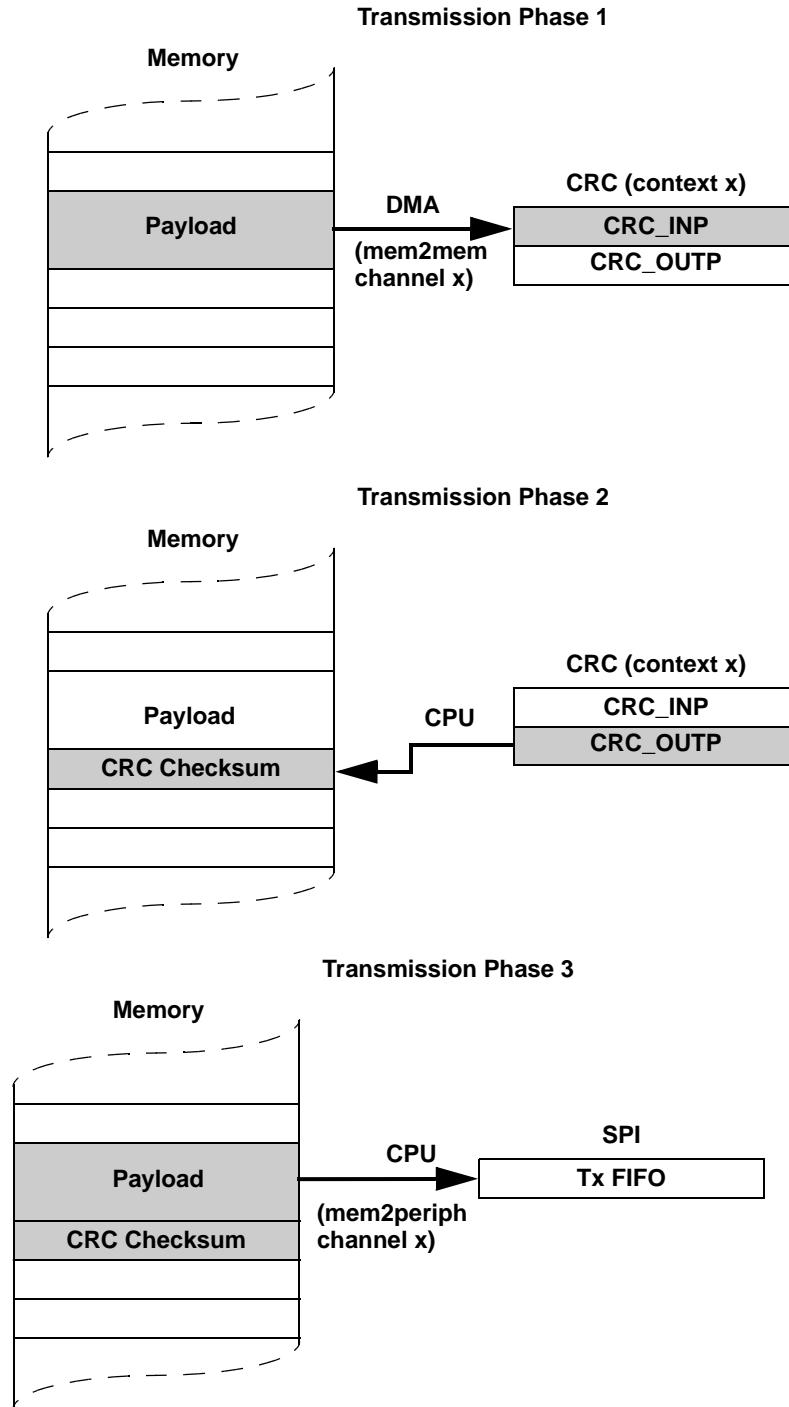


Figure 431. DMA-CRC Transmission Sequence

The following sequence, as given in [Figure 432](#), shall be applied to manage the reception data flow:

- DMA/CRC module configuration (context x, channel x) by CPU

Cyclic Redundancy Check (CRC)

- Data block (payload + CRC) transfer from the PERIPH (e.g., SPI Rx fifo) module to the MEM (phase 1) by DMA (periph2mem data transfer, channel x)
- Data block transfer (payload + CRC) transfer from the MEM to the CRC module (CRC_INP register) to calculate the CRC signature (phase 2) by DMA (mem2mem data transfer, channel x)
- CRC signature check from the CRC module (CRC_OUTP register) by CPU (phase 3)

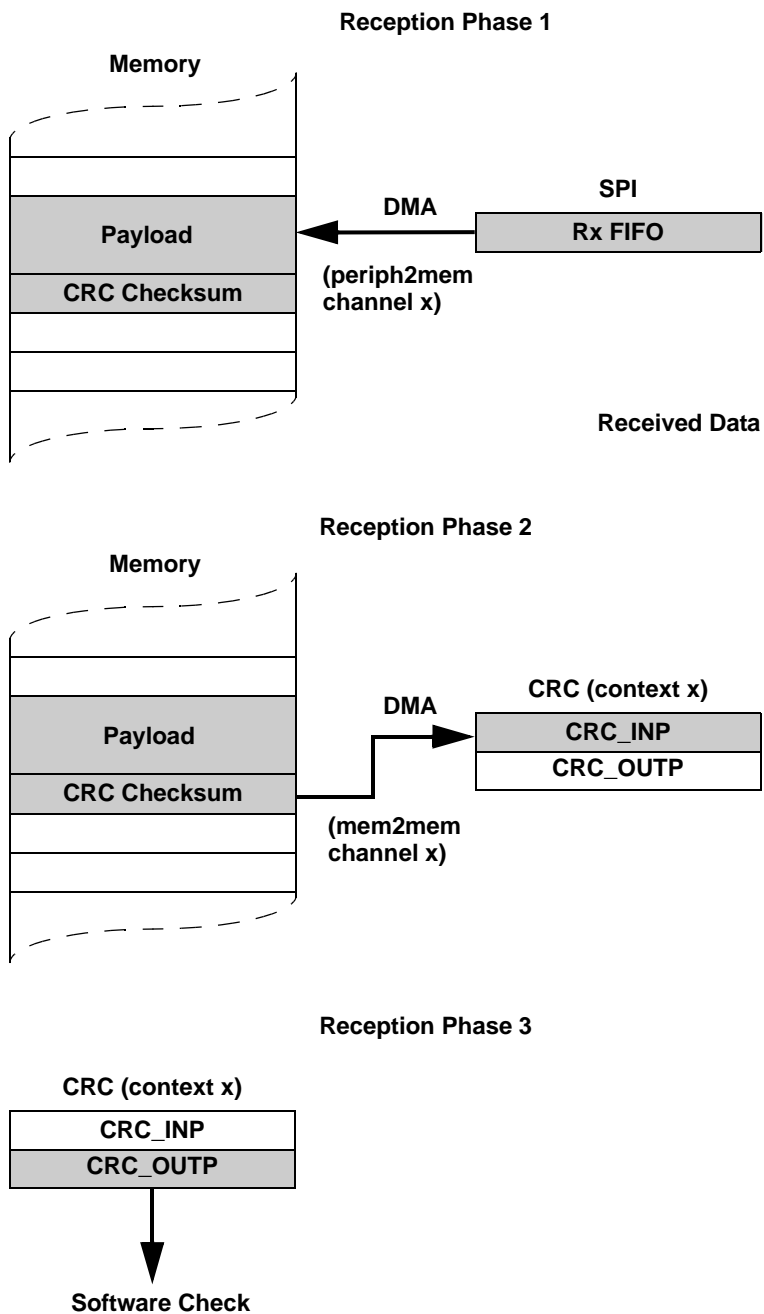


Figure 432. DMA-CRC Reception Sequence

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter 34

Boot Assist Module (BAM)

This chapter describes the Boot Assist Module (BAM).

34.1 Overview

The Boot Assist Module is a block of read-only one-time programmed memory containing VLE code that is executed according to the boot mode of the device.

The BAM allows downloading boot code via the FlexCAN or LINFlex interfaces into internal SRAM and then executing it.

34.2 Features

The BAM provides the following features:

- Boot from Internal Code Flash. Selected as default (using internal pull down on FAB pin).
- MPC5604E in static mode if internal flash is not initialized or invalid
- Programmable 64-bit password protection for serial boot mode
- Serial boot loads the application boot code from a FlexCAN or LINFlex bus into internal SRAM. BAM can accept a password via the used serial communication channel to grant the legitimate user access to the non-volatile memory.
- Censorship protection for internal flash module

34.3 Boot modes

The MPC5604E device supports the following boot modes:

- Single Chip (SC) — The device boots from the first bootable section of the Flash main array.
- Serial Boot (SBL) — The device downloads boot code from either LINFlex or FlexCAN interface and then execute it.

If booting is not possible with the selected configuration (e.g., if no Boot ID is found in the selected boot location) then the device enters the static mode.

34.4 Memory map

The BAM code resides in a reserved 8 KB ROM mapped from address 0xFFFF_C000. The address space and memory used by the BAM application is shown in [Table 441](#).

Table 441. BAM memory organization

Parameter	Address
BAM entry point	0xFFFF_C000
Downloaded code base address	0x4000_0100

The RAM location where to download the code can be any 4-byte-aligned location starting from the address 0x4000_0100.

34.5 Functional description

34.5.1 Entering boot modes

The MPC5604E detects the boot mode based on external pins and device status. The following sequence applies (see [Figure 433](#)):

- To boot either from FlexCAN or LINFlex, the device must be forced into an Alternate Boot Loader Mode via the FAB (Force Alternate Boot Mode), which must be asserted before initiating the reset sequence. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pins (see [Table 442](#)).
- If FAB is not asserted, the device boots from the lowest Flash sector that contains a valid boot signature.
- If no Flash sector contains a valid boot signature, the device will go into static mode.

Figure 433. Alternate Boot Mode Selection

Table 442. Hardware configuration to select boot mode

FAB	ABS[1:0]	Standby-RAM Boot Flag	Boot ID	Boot Mode
1	00	0	—	LINFlex without autobaud
1	01	0	—	FlexCAN without autobaud
1	10	0	—	Autobaud scan

34.5.2 Reset Configuration Half Word (RCHW)

The MPC5604E Flash is partitioned into boot sectors as shown in [Table 444](#).

Each boot sector contains the Reset Configuration Half-Word (RCHW) at offset 0x00.

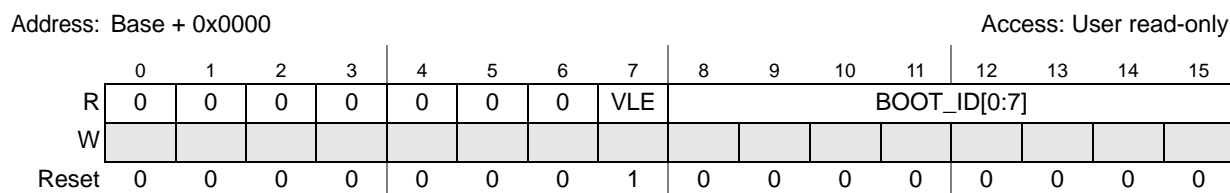


Figure 434. Reset Configuration Half Word (RCHW)

Table 443. RCHW field descriptions

Field	Description
0-6	Reserved
7 VLE	VLE Indicator This bit configures the MMU for the boot block to execute as either Classic Power Architecture Book E code or as Freescale VLE code.
8-15 BOOT_ID[0:7]	Is valid if its value is 0x5A, then the sector is considered bootable.

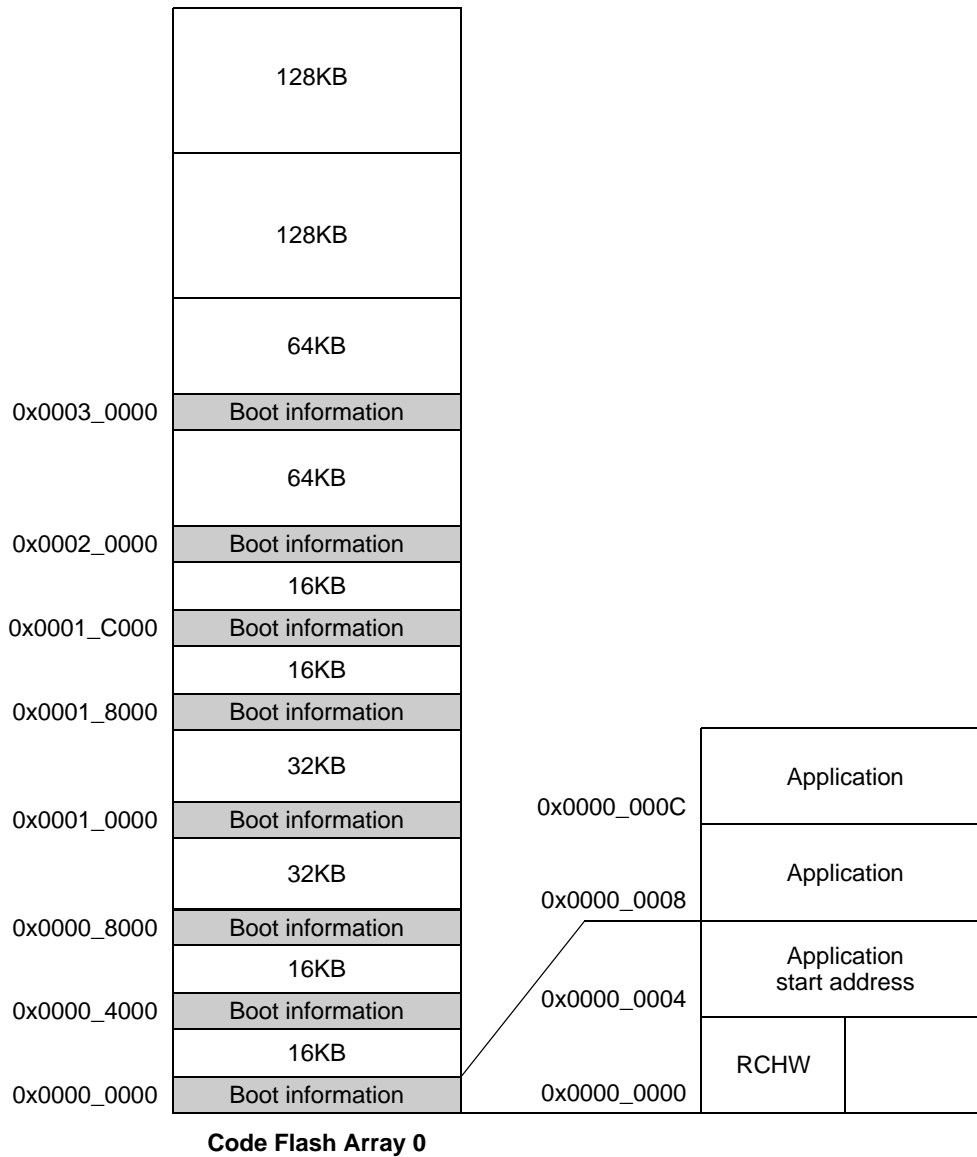


Table 444. Flash boot sector

Block	Address
0	0x0000_0000
1	0x0000_4000
2	0x0000_8000
3	0x0001_0000
4	0x0001_8000
5	0x0001_C000
6	0x0002_0000
7	0x0003_0000

34.5.3 Single chip boot mode

In single chip boot mode, the hardware searches the flash boot sector for a valid boot ID. As soon the device detects a bootable sector, it jumps within this sector and reads the 32-bit word at offset 0x4. This word is the address where the startup code is located (reset boot vector).

Then the device executes this startup code. A user application should have a valid instruction at the reset boot vector address.

If a valid RCHW is not found, the BAM code is executed. In this case BAM moves the MPC5604E into static mode.

34.5.3.1 Boot and alternate boot

Some applications require an alternate boot sector in the flash so that the main sector in flash can be erased and reprogrammed in the field.

When an alternate boot is needed, the user can create two bootable sectors. The low sector is the main boot sector and the high sector is the alternate boot sector. The alternate boot sector does not need to be consecutive to the main boot sector.

This ensures that even if one boot sector is erased still there will always be another active boot sector:

- Sector shall be activated (i.e., program a valid BOOT_ID instead of 0xFF as initially programmed).
- Sector shall be deactivated writing to 0 some of the bits BOOT_ID bit field (bit1 and/or bit3, and/or bit4, and/or bit6).

34.5.4 Boot through BAM

34.5.4.1 Executing BAM

Single chip boot mode is managed by hardware and BAM does not participate in it.

BAM is executed only on these two following cases:

- Serial boot mode has been selected by FAB pin
- Hardware has not found a valid Boot-ID in any Flash boot locations

If one of these conditions is true, the device fetches code at location 0xFFFF_C000 and BAM application starts.

34.5.4.2 BAM software flow

Figure 436 illustrates the BAM logic flow.

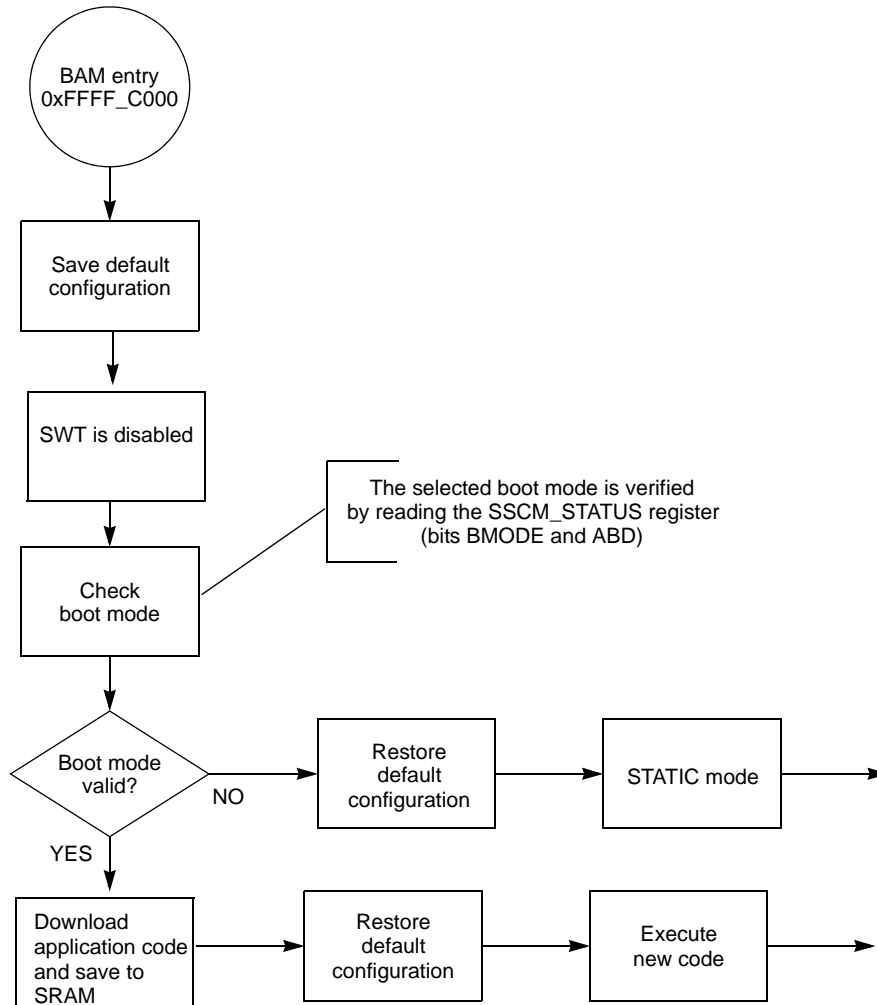


Figure 436. BAM logic flow

The first action is to save the initial device configuration. The next step is to disable the software watchdog. In this way, it is possible to restore the initial configuration after downloading the new code but before executing it. This allows the new code being executed as the device was just coming out of reset.

The BMODE and ABD fields of SSCM_STATUS register (see [Section 12.3.1.1, “System Status Register”](#)) indicate which boot has to be executed (see [Table 445](#)).

If the BMODE field shows either a single chip value (011) or a reserved value, the boot mode is not considered valid and the BAM pushes the device into static mode.

In all other cases the code of the relative boot is called. Data is downloaded and saved into proper SRAM location.

Table 445. Fields of SSCM STATUS register used by BAM

Field	Description
BMODE [2:0]	Device Boot Mode 000 Autobaud boot 001 CAN Serial Boot Loader 010 SCI Serial Boot Loader 011 Single Chip 100–111 Reserved This field is updated only during destructive reset.

Then, the initial device configuration is restored and the code jumps to the address of downloaded code. At this point BAM has just finished its task.

If an error occurs, (e.g., communication error, wrong boot selected, etc.), the BAM restores the default configuration and puts the device into static mode. Static mode means the device enters the low power mode SAFE and the processor executes a wait instruction. This is needed if the device cannot boot in the selected mode. During BAM execution and after, the mode reported by the field S_CURRENT_MODE of the register ME_GS in the module ME Module is “DRUN”.

NOTE

The device exits the Static mode only after Reset.

34.5.4.3 BAM resources

BAM uses/initializes the following MCU resources:

- ME and CGM modules to initialize mode and clock sources
- CAN_0, LINFlex_0, and their pads when performing serial boot mode
- SSCM to check the boot mode and during password check (see [Table 445](#) and [Figure 437](#))
- External oscillator

The following hardware resources are used only when autobaud feature is selected:

- STM to measure the baud rate
- CMU to measure the external clock frequency related to the internal RC clock source
- FMPLL to work with system clock near the maximum allowed frequency (this to have higher resolution during baud rate measurement).

As already mentioned, the initial configuration is restored before executing the downloaded code.

When the autobaud feature is disabled, the system clock is selected directly from the external oscillator. Thus the oscillator frequency defines baud rates for serial interfaces used to download the user application (see [Table 446](#)).

Table 446. Serial boot mode without autobaud—baud rates

Crystal frequency (MHz)	LINFlex baud rate (baud)	FlexCAN bit rate (bit/s)
f_{extal}	$f_{\text{extal}} / 833$	$f_{\text{extal}} / 40$
8	9600	200 K
12	14400	300 K
16	19200	400 K
20	24000	500 K
40	48000	1 M

34.5.4.4 Download and execute the new code

From a high level perspective, the download protocol follows these steps:

1. Send message and receive acknowledge message for autobaud or autobit rate selection. (optional step).
2. Send 64-bit password.
3. Send start address, size of downloaded code in bytes, and VLE bit¹.
4. Download data.
5. Execute code from start address.

Each step must be complete before the next step starts.

These steps are correct if autobaud is disabled. Otherwise, to measure the baud rate, some data is sent from the host to the MCU before step 1 (see [Section 34.6.1, “Autobaud feature”](#)).

The communication is done in half duplex manner. Any transmission from the host is followed by the MCU transmission:

1. Host sends data to MCU and start waiting.
2. MCU echoes to host the data received.
3. The host verifies if echo is correct.
 - If data is correct, the host can continue to send data.
 - If data is not correct, the host stops transmitting and the MCU needs to be reset.

All multi-byte data structures are sent MSB first.

A more detailed description of these steps follows.

34.5.4.5 Download 64-bit password and password check

The first 64 received bits represent the password. This password is sent to the Password Check procedure for verification.

Password check data flow is shown in [Figure 437](#) where:

1. Since the device supports only VLE code and does not support Book E code, this flag is used only for backward compatibility.

Boot Assist Module (BAM)

- SSCM_STATUS[SEC] = 1 means flash secured
- SSCM_STATUS[PUB] = 1 means flash with public access.

In case of flash with public access, the received password is compared with the public password 0xFEED_FACE_CAFE_BEEF.

If public access is not allowed but the flash is not secured, the received password is compared with the value saved on NVPWD0 and NVPWD1 registers.

In uncensored devices, it is possible to download code via LINFlex or FlexCAN (serial boot mode) into internal SRAM with any 64-bit private password stored in the flash and provided during the boot sequence.

In the previous cases, comparison is done by the BAM application. If it goes wrong, BAM pushes the device into static mode.

In case of public access not allowed and flash secured, the password is written into SSCM[PWCMPH/L] registers.

In case of flash secured with public access not allowed (user password configured in the NVPWD0 and NVPWD1 registers), the user must set the swapped password: NVPWD1 and NVPWD0 to access the device (refer to following examples).

Example 1.

In devices with flash secured, registers are programmed:

```
NVPWD0 = 0x87654321
NVPWD1 = 0x12345678
NVSCI0  = 0x55AA1111
NVSCI1  = 0x55AA1111
```

To download the code via SLB, the provided password is 0x1234_5678_8765_4321 (swapped WITH respect to NVPWD0/NVPWD1 → NVPWD1/NVPWD0).

Example 2.

In devices with flash NOT secured, registers are programmed:

```
NVPWD0 = 0x87654321
NVPWD1 = 0x12345678
NVSCI0  = 0x55AA55AA
NVSCI1  = 0x55AA55AA
```

To download the code via SLB the provided password is 0x8765_4321_1234_5678 (as expected from NVPWD0/NVPWD1).

After a fixed time waiting, comparison is done by hardware. Then BAM again verifies the SEC flag in SSCM_STATUS:

- SEC = 0, flash is now unsecured and BAM continues its task
- SEC = 1, flash is still secured because password was wrong; BAM puts the device into standby mode.

This fixed time depends on external crystal oscillator frequency (XOSC). With XOSC of 12 MHz, fixed time is 350 ms.

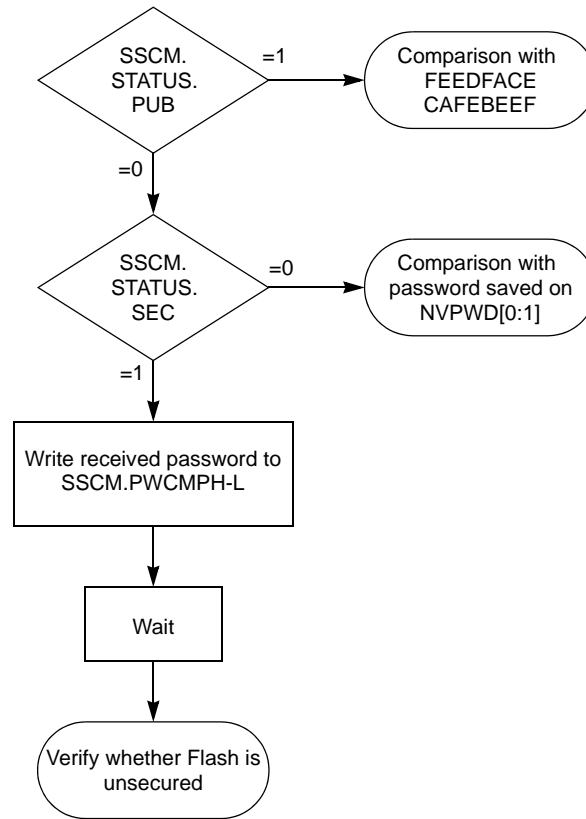


Figure 437. Password check flow

34.5.4.6 Download start address, VLE bit and code size

The next 8 bytes received by the MCU contain a 32-bit Start Address, the VLE mode bit and a 31-bit code Length as shown in Figure 438.

The VLE bit (Variable Length Instruction) indicates which instruction set for which the code has been compiled. This device family supports only VLE = 1. The bit is used for backward compatibility.

The Start Address defines where the received data will be stored and where the MCU will branch after the download is finished. The two LSB bits of the Start Address are ignored by the BAM program, such that the loaded code should be 32-bit word aligned.

The Length defines how many data bytes have to be loaded.

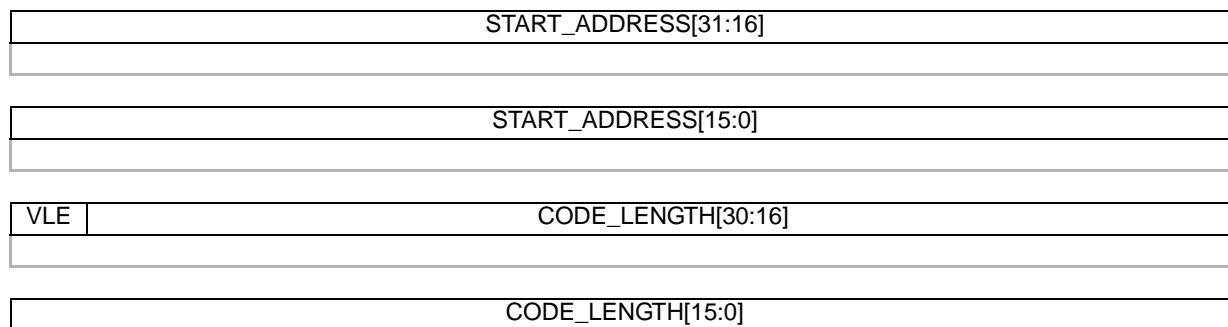


Figure 438. Start address, VLE bit and download size in bytes

34.5.4.7 Download data

Each byte of data received is stored into device’s SRAM, starting from the address specified in the previous protocol step.

The address increments until the number of bytes of data received matches the number of bytes specified in the previous protocol step.

Since the SRAM is protected by 32-bit wide Error Correction Code (ECC), BAM always writes bytes into SRAM grouped into 32-bit words. If the last byte received does not fall onto a 32-bit boundary, BAM fills it with 0 bytes.

Then a “dummy” word (0x0000_0000) is written to avoid ECC error during core prefetch.

34.5.4.8 Execute code

The BAM program waits for the last echo message transmission being completed.

Then it restores the initial MCU configuration and jumps to the loaded code at Start Address that was received in step 2 of the protocol.

At this point BAM has finished its tasks and MCU is controlled by new code executing from SRAM.

34.5.5 Boot from UART—autobaud disabled

34.5.5.1 Configuration

Boot from UART protocol is implemented by LINFlex_0 module. Pins used are:

- LINFlex_TX corresponds to pad [18]
- LINFlex_RX corresponds to pad [19]

When autobaud feature is disabled, the system clock is driven by external oscillator.

LINFlex controller is configured to operate at a baud rate = system clock frequency/833 (see [Table 446](#) for baud rate example), using 8-bit data frame without parity bit and 1 stop bit.

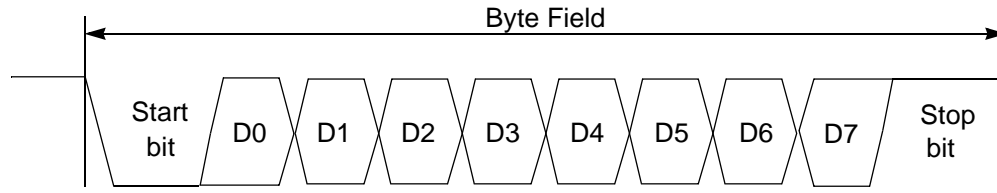


Figure 439. LINFlex bit timing in UART mode

34.5.5.2 UART boot mode download protocol

Table 447 summarizes the download protocol and BAM action during the UART boot mode.

Table 447. UART boot mode download protocol (autobaud disabled)

Protocol step	Host sent message	BAM response message	Action
1	64-bit password (MSB first)	64-bit password	Password checked for validity and compared against stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + 31-bit number of bytes (MSB first)	VLE bit + 31-bit number of bytes (MSB first)	Size of download is stored for future use. Verify if VLE bit is set to 1.
4	8 bits of raw binary data	8 bits of raw binary data	8-bit data are packed into 32-bit word. This word is saved into SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step.
5	none	none	Branch to downloaded code.

34.5.6 Bootstrap with FlexCAN—autobaud disabled

34.5.6.1 Configuration

Boot from FlexCAN protocol is implemented by the FlexCAN_0 module. Pins used are:

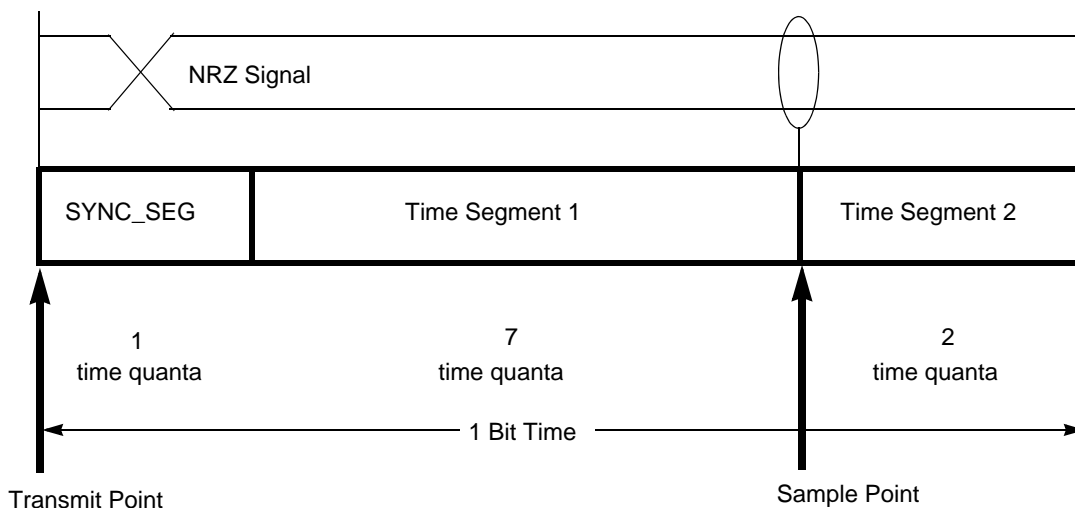
- CAN_TX corresponds to pad[16]
- CAN_RX corresponds to pad[17]

Boot from FlexCAN with autobaud disabled uses system clock driven by the external oscillator.

The FlexCAN controller is configured to operate at a baud rate equal to the system clock frequency/40 (see Table 446 for examples of baud rate).

It uses the standard 11-bit identifier format detailed in the FlexCAN 2.0A specification.

FlexCAN controller bit timing is programmed with 10 time quanta, and the sample point is 2 time quanta before the end, as shown in Figure 440.



1 time Quanta = 4 system clock periods

Figure 440. FlexCAN bit timing

34.6 FlexCAN boot mode download protocol

Table 448 summarizes the download protocol and BAM action during the FlexCAN boot mode. All data are transmitted byte-wise.

Table 448. FlexCAN boot mode download protocol (autobaud disabled)

Protocol step	Host sent message	BAM response message	Action
1	FlexCAN ID 0x011 + 64-bit password	FlexCAN ID 0x001 + 64-bit password	Password checked for validity and compared against stored password.
2	FlexCAN ID 0x012 + 32-bit store address + VLE bit + 31-bit number of bytes	FlexCAN ID 0x002 + 32-bit store address + VLE bit + 31-bit number of bytes	Load address is stored for future use. Size of download is stored for future use. Verify if VLE bit is set to 1.
3	FlexCAN ID 0x013 + 8 to 64 bits of raw binary data	FlexCAN ID 0x003 + 8 to 64 bits of raw binary data	8-bit data are packed into 32-bit words. These words are saved into SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step.
4	none	none	Branch to downloaded code.

34.6.1 Autobaud feature

The autobaud feature allows boot operation with a wide range of baud rates independent of the external oscillator frequency.

34.6.1.1 Configuration

Baud rate is measured by BAM using the System Timer Module (STM), which is driven by the system clock.

One of the main differences with the previous cases is that system clock is not driven directly by the external oscillator. Instead, it is driven by the FMPLL output. The reason is that to have an optimum baud rate measurement resolution, system clock needs to be nearer to the maximum allowed device’s frequency. This is achieved with the following two steps:

- Using the Clock Monitor Unit (CMU) and the RC oscillator, the external frequency is roughly measured.
- Using the FMPLL, the system clock is configured to be close to but lower than the maximum allowed frequency.

The relation between system clock frequency and external clock frequency with FMPLL configuration value is shown in [Table 449](#).

Table 449. System clock frequency related to external clock frequency

f_{osc} [MHz]	f_{rc}/f_{osc} ¹	f_{sys} [MHz]
4–8	4–2	16–32
8–12	2–4/3	32–48
12–16	4/3–1	36–48
16–24	1–2/3	32–48
> 24	< 2/3	> 24

¹ These values and consequently the f_{sys} suffer from the precision of RC internal oscillator used to measure f_{osc} through CMU module.

34.6.1.2 Boot from UART with autobaud enabled

The only difference between booting from UART with autobaud enabled and booting from UART with autobaud disabled is that a further byte is sent from the host to the MCU when autobaud is enabled. The value of that byte is 0x00.

This first byte measures the time from falling edge and rising edge. The baud rate can be calculated from this time.

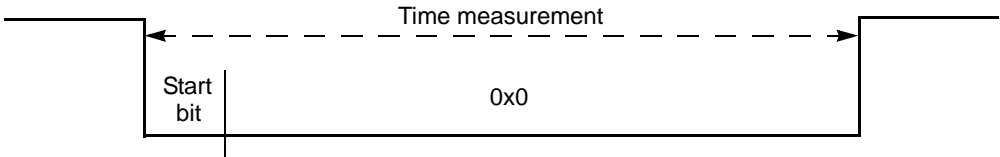


Figure 441. Baud measurement on UART boot

Initially the UART RX pin is configured as GPIO input and it waits in polling for the first falling edge, then STM starts. UART RX pin waits again for the first rising. Then STM stops and from its measurement baud rate is computed.

The LINFlex module is configured to work in UART mode with the calculated baud rate. Then an acknowledge byte (0x59, ASCII char “Y”) is sent.

From this point, the BAM follows the normal UART mode boot protocol (see [Figure 442](#)).

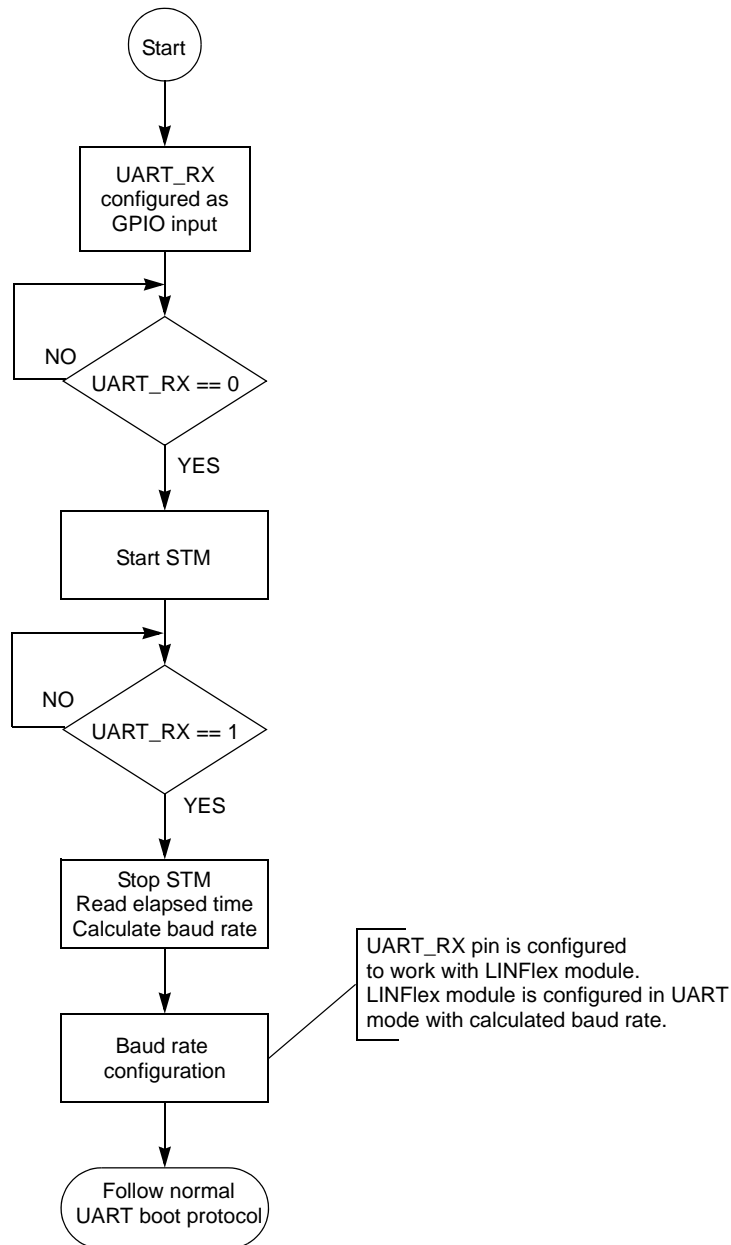


Figure 442. BAM rate measurement flow during UART boot

34.6.1.2.1 Choosing the host baud rate

The calculation of the UART baud rate from the length of the first 0 byte that is received, allows the operation of the boot loader with a wide range of baud rates. However, to ensure proper data transfer, the upper and lower limits have to be kept.

SCI autobaud rate feature operates by polling the LINFlex_RX pin for a low signal. The length of time until the next low to high transition is measured using the System Timer Module (STM) time base. This high-low-high transition is expected to be a zero byte: a start bit (low) followed by eight data bits (low) followed by a stop bit (high).

Upon reception of a zero byte and configuration of the baud rate, an acknowledge byte is returned to the host using the selected baud rate.

Time base is enabled at reception of first low bit, disabled and read at reception of next high bit. Error introduced due to polling will be small (typically < 6 cycles).

The following equation gives the relation between baud rate and LINFlex register configuration:

$$LDIV = \frac{F_{cpu}}{16 \cdot baudrate} \quad \text{Eqn. 1}$$

LDIV is an unsigned fixed point number and its mantissa is coded into 13 bits of the LINFlex's register LINIBRR.

From this equation and considering that a single UART transmission contains 9 bits, it is possible to obtain the connection between time base measured by STM and LINIBB register:

$$LINIBRR = \frac{timebase}{144} \quad \text{Eqn. 2}$$

To minimize errors in baud rate, a large external oscillator frequency value and low baud rate signal are preferred.

Example 3. Baud rate calculation for 24 kBaud signal

Considering a 24 kbaud signal and the device operating with 20 MHz external frequency.

Over 9 bits the STM will measure: $(9 \times 20 \text{ MHz})/24 \text{ kbaud} = 7497$ cycles.

Error expected to be approximately ± 6 cycles due to polling.

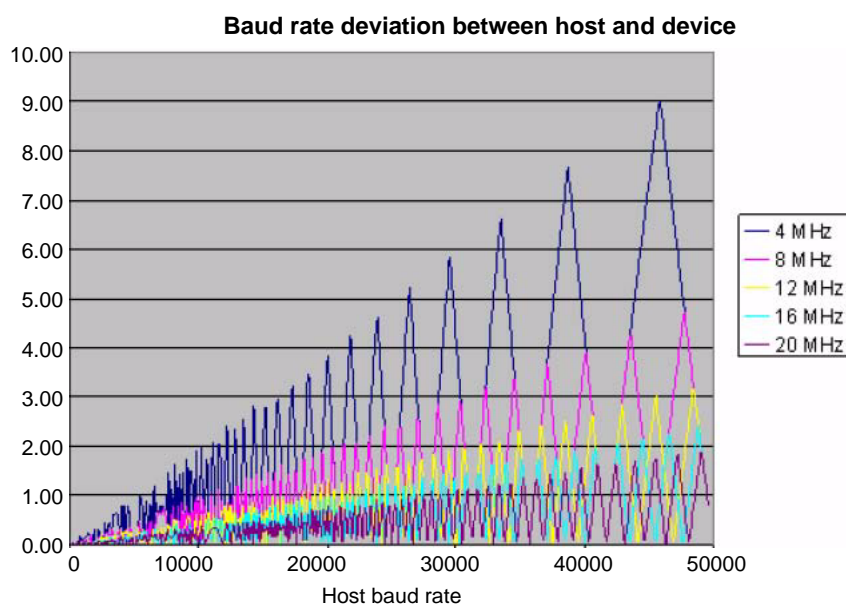
Thus, LINIBB will be set to 52 (rounding required). This results in a baud rate of 24.038 kbaud. Error of < 0.2%.

To maintain the maximum deviation between host and calculated baud rate, recommendations for the user are listed [Table 450](#).

Table 450. Maximum and minimum recommended baud rates

$f_{\text{sys}} = f_{\text{xtal}}$ (MHz)	Max baud rate for guaranteed < 2.5% deviation	Min baud rate for guaranteed < 2.5% deviation
4	13.4 Kbit/s (SBR = 19)	30 bit/s (SBR = 8192)
8	26.9 Kbit/s (SBR = 19)	60 bit/s (SBR = 8192)
12	38.4 Kbit/s (SBR = 20)	90 bit/s (SBR = 8192)
16	51.2 Kbit/s (SBR = 20)	120 bit/s (SBR = 8192)
20	64.0 Kbit/s (SBR = 20)	150 bit/s (SBR = 8192)

Higher baud rates high may be used, but the user will be required to ensure they fall within an acceptable error range. This is illustrated in [Figure 443](#), which shows the effect of quantization error on the baud rate selection.


Figure 443. Baud rate deviation between host and MPC5604E

Example 4. Baud rate calculation for 250 kBaud signal

Considering reception of a 250kBaud signal from the host and MPC5604E operating with a 4 MHz oscillator. Over 9 bits the time base will measure: $(9 \times 4e6)/250e3 = 144$ cycles.

Thus, LINIBB is set to $144/144 = 1$. This results in a baud rate of exactly 250 kBd.

However, a slower 225 kBd signal operating with 4 MHz XTAL would again result in $LINIBB = 1$, but this time with an 11.1% deviation.

34.6.1.3 Boot from FlexCAN with autobaud enabled

The only difference between booting from FlexCAN with autobaud enabled and booting from FlexCAN with autobaud disabled is that the following initialization FlexCAN frame is sent for baud measurement purposes from the host to the MCU when autobaud is enabled:

- Standard identifier = 0x0,
- Data Length Code (DLC) = 0x0.

As all the bits to be transmitted are dominant bits, there is a succession of 5 dominant bits and 1 stuff bit on the FlexCAN network (see [Figure 444](#)).

From the duration of this frame, the MCU calculates the corresponding baud rate factor with respect to the current CPU clock and initializes the FlexCAN interface accordingly.

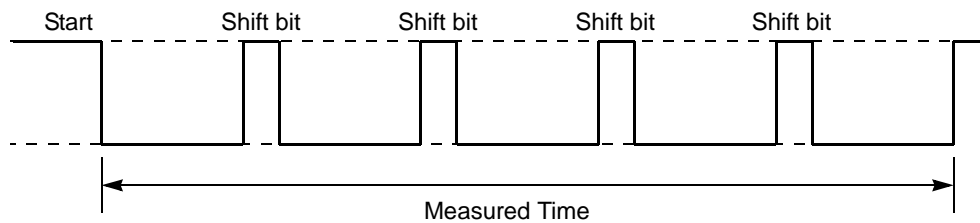


Figure 444. Bit time measure

In UART boot mode, the FlexCAN RX pin is first configured to work as a GPIO input. In the end of baud rate measurement, it switches to work with the FlexCAN module.

The baud rate measurement flow is detailed in [Figure 445](#).

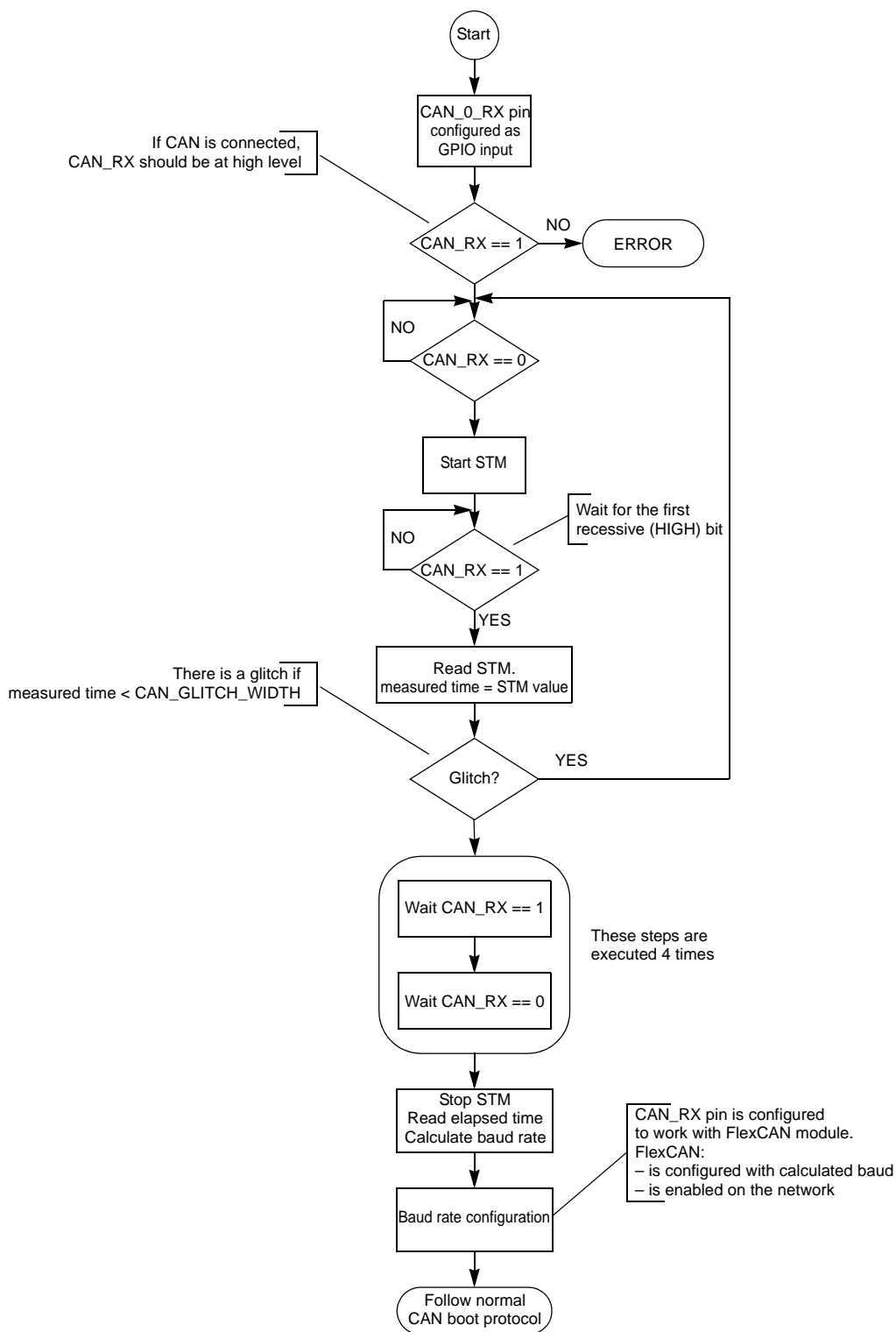


Figure 445. BAM rate measurement flow during FlexCAN boot

34.6.1.3.1 Choosing the host baud rate

The calculation of the FlexCAN baud rate allows the operation of the boot loader with a wide range of baud rates. However, to ensure proper data transfer, the upper and lower limits have to be kept.

Pins are measured until reception of the 5th recessive bit. Thus a total of 29 bit-times are measured (see [Figure 444](#)).

When calculating bit times and prescalers, to minimize any errors, ideally operate with the minimum system clock prescaler divider (CAN_CR[PRES DIV]) and maximum number of time quanta possible.

After measuring the 29 bit times, the results stored in the STM time base are used to select PRES DIV. The number of time quanta in a FlexCAN bit time is given by:

$$\text{Bit_time} = \text{SYNCSEG} + \text{TSEG1} + \text{TESG2}$$

SYNCSEG = Exactly one time quantum.

$$\text{TSEG1} = \text{PROGPSEG} + \text{PSEG1} + 2$$

$$\text{TSEG2} = \text{PSEG2} + 1$$

$$\text{Time base result} = 29 \times (\text{Presdiv}+1) \times (\text{SYNCSEG} + \text{TSEG1} + \text{TSEG2})$$

FlexCAN protocol specifies that the FlexCAN bit timing should comprise a minimum of 8 time quanta and a maximum of 25 time quanta. Therefore, the available range is:

$$8 \leq 1 + \text{TSEG1} + \text{TESG2} \leq 25$$

For 29 bit times, the possible range in which the result in the time base may lie, accounting for PRES DIV, is:

$$(232 \times (1 + \text{PRES DIV})) \leq \text{time base} \leq (725 \times (1 + \text{PRES DIV}))$$

Therefore, the available values of the time base can be divided into windows of 725 counts.

Table 451. Prescaler/divider and time base values

PRES DIV	Time base Minimum	Time base Maximum
0	232	725
1	726	1450
2	1451	2175
3	2176	2900

In the BAM, the time base is divided by 726, the remainder is discarded. The result provides the CAN_CR[PRES DIV] to be selected.

To help compensate for any error in the calculated baud rate, the resynchronization jump width will be increased from its default value of 1 to a fixed value of 2 time quanta. This is the maximum value allowed that can accommodate all permissible can baud rates. See [Table 452](#).

Timing segment 2 is kept as large as possible to keep sample time within bit time.

Worked examples showing FlexCAN Autobaud rate:

Example 5. 8 MHz crystal

Consider case where using an 8 MHz crystal, user attempts to send 1 MB (max permissible baud rate) FlexCAN message.

- Time base, clocking at crystal frequency, would measure:
- $1\text{MB} = 8 \text{ clocks/bit} \Rightarrow 29 * 8 = 232 \text{ clocks}$
- To calculate $\text{PRES DIV} = 232/725 \Rightarrow \text{PRES DIV} = 0$
- To calculate time quanta requirement:
- Time base result = $29 * (\text{Presdiv}+1) * (\text{SYNCSEG} + \text{TSEG1} + \text{TSEG2})$
- $232 = 29 * 1 * (1 + \text{TSEG1} + \text{TSEG2})$
- $1 + \text{TSEG1} + \text{TSEG2} = 8.$
- From the lookup table, $\text{CANA_CR} = 0x004A_2001.$
- This give a baud rate of X. This give 0% error.

Example 6. 20 MHz crystal

Consider case where using a 20 MHz crystal, user attempts to send 62.5 Kb/s FlexCAN message.

- Time base, clocking at crystal frequency, would measure:
- $62.5 \text{ Kb/s} = 320 \text{ clocks/bit} \Rightarrow 29 * 320 = 9280 \text{ clocks}$
- To calculate $\text{PRES DIV} = 9280/725 = 12r580 \Rightarrow \text{PRES DIV} = 12$
- To calculate time quanta requirement:
- Time base result = $29 * (\text{Presdiv}+1) * (\text{SYNCSEG} + \text{TSEG1} + \text{TSEG2})$
- $9280 = 29 * 13 * (1 + \text{TSEG1} + \text{TSEG2})$
- $1 + \text{TSEG1} + \text{TSEG2} = 24.6 \sim 25$ (need to round up - S/W must track remainder)
- From the lookup table, $\text{CANA_CR} = 0x0C7F_200D.$
- This give a baud rate of 61.538k Baud
- This equates to an error of: $\sim 1.6\%$

This excludes cycle count error introduced due to software polling likely to be ~ 6 system clocks.

34.6.2 Interrupt

No interrupts are generated by or are enabled by the BAM.

Chapter 35

Inter-Integrated Circuit Bus Controller Module (I2C)

35.1 Introduction

35.1.1 Overview

The Inter-Integrated Circuit (I²C™ or IIC) bus is a two wire bidirectional serial bus that provides a simple and efficient method of data exchange between devices. It minimises the number of external connections to devices and does not require an external address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of module clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400pF.

35.1.2 Features

The I²C module has the following key features:

- Compatible with I²C Bus standard
- Up to 100 kbps with maximum bus loading
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Programmable slave address and glitch input filter
- Interrupt driven byte-by-byte data transfer
- Software selectable acknowledge bit
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

35.1.3 Modes of Operation

The I²C functions the same in normal, special, and emulation modes. It has two low power modes: doze and stop.

- Run Mode
This is the basic mode of operation.
- Stop Mode
This is the lowest power saving mode and allows the system to turn of all the clocks to the I²C module. This state can only be entered when there are no active transfers on the bus.

35.1.4 Block Diagram

The block diagram of the I²C module is shown in [Figure 446](#)

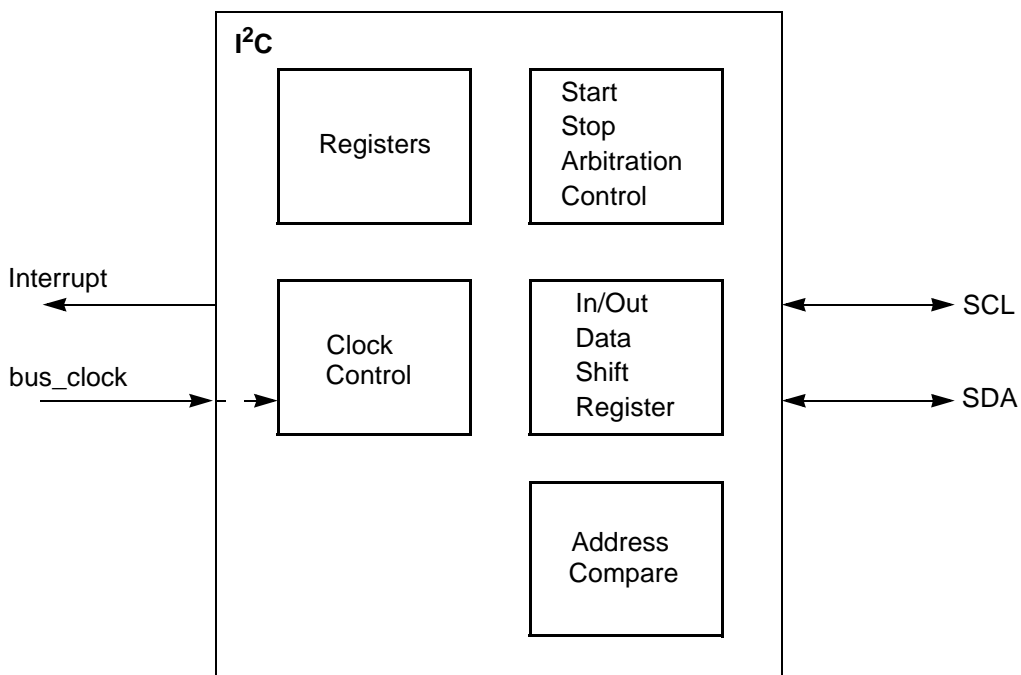


Figure 446. I²C Block Diagram

35.2 External Signal Description

35.2.1 Overview

The Inter-Integrated Circuit (I²C) module has 2 external pins.

35.2.2 Detailed Signal Descriptions

35.2.2.1 SCL

This is the bidirectional Serial Clock Line (SCL) of the module, compatible with the I²C-Bus specification.

35.2.2.2 SDA

This is the bidirectional Serial Data line (SDA) of the module, compatible with the I²C-Bus specification.

35.3 Memory Map/Register Definition

35.3.1 Overview

This section provides a detailed description of all memory-mapped registers in the I²C module.

35.3.2 Module Memory Map

The memory map for the I²C module is given below in [Table 452](#). The total address for each register is the sum of the base address for the I²C module and the address offset for each register.

NOTE

Please see the device user guide for base address of this module.

Table 452. Module Memory Map

Address	Register	Size	Access	Mode ¹
Base + 0x00	I ² C Bus Address Register (IBAD)	8	R/W	A
Base + 0x01	I ² C Bus Frequency Divider Register (IBFD)	8	R/W	A
Base + 0x02	I ² C Bus Control Register (IBCR)	8	R/W	A
Base + 0x03	I ² C Bus Status Register (IBSR)	8	R/W	A
Base + 0x04	I ² C Bus Data I/O Register (IBDR)	8	R/W	A
Base + 0x05	I ² C Bus Interrupt Config Register (IBIC)	8	R/W	A
Base + 0x06	Unused	8	R	A
Base + 0x07	Unused	8	R	A
Base + 0x08 – Base + 0x3FFF	Reserved		See Note ²	

¹ **U** = User Mode, **S** = Supervisor Mode, **A** = All (No restrictions)

² If enabled at the SoC level, reads or writes to these registers will cause bus aborts. Refer to the System Services Module documentation for more details.

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the IBDF

register for the frequency divider is accessible by a 16-bit READ/WRITE to address Base + 0x000, but performing a 16-bit access to Base + 0x001 is illegal.

35.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

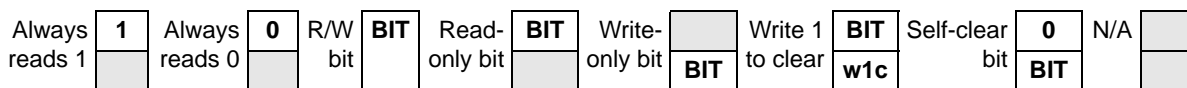


Figure 447. Key to Register Fields

35.3.3.1 I²C Address Register

This register contains the address the I²C Bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

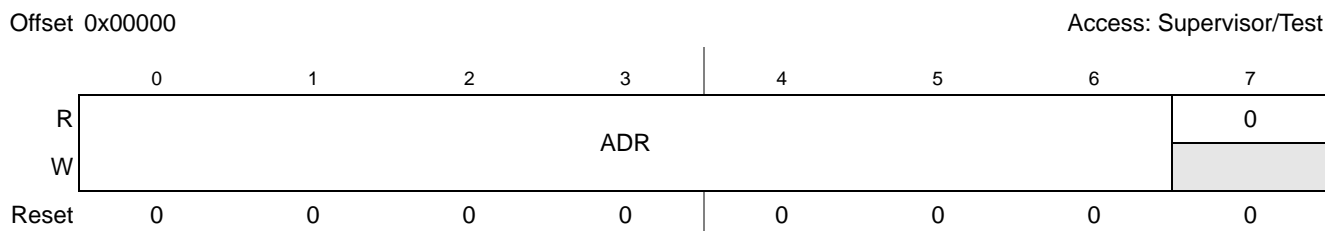


Figure 448. I²C Bus Address Register (IBAD)

Table 453. IBAD Field Descriptions

Field	Description
7–1 ADR	Slave Address. Specific slave address to be used by the I ² C Bus module. Note: The default mode of I ² C Bus is slave mode for an address match on the bus.
0	Reserved, should be cleared; will always read 0.

35.3.3.2 I²C Frequency Divider Register

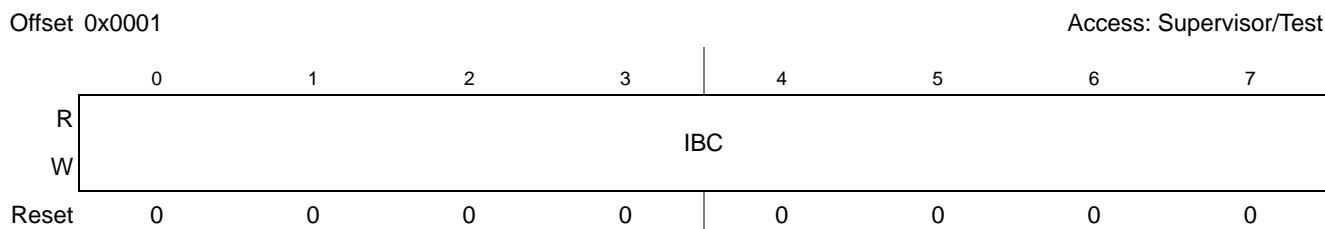


Figure 449. I²C Bus Frequency Divider Register (IBFD)

Table 454. IBFD Field Descriptions

Field	Description
7–0 IBC	I-Bus Clock Rate. This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows: 7–6 select the prescaled shift register (see Table 455) 5–3 select the prescaler divider (see Table 456) 2–0 select the shift register tap point (see Table 457)

Table 455. I-Bus Multiplier Factor

IBC7–6	MUL
00	01
01	02
10	04
11	RESERVED

Table 456. I-Bus Prescaler Divider Values

IBC5–3	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 457. I-Bus Tap and Prescale Values

IBC2-0	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3

Table 457. I-Bus Tap and Prescale Values

IBC2-0	SCL Tap (clocks)	SDA Tap (clocks)
110	12	4
111	15	4

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 456. All subsequent tap points are separated by 2^{IBC5-3} as shown in the tap2tap column in Table 456. The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA i.e. the SDA hold time.

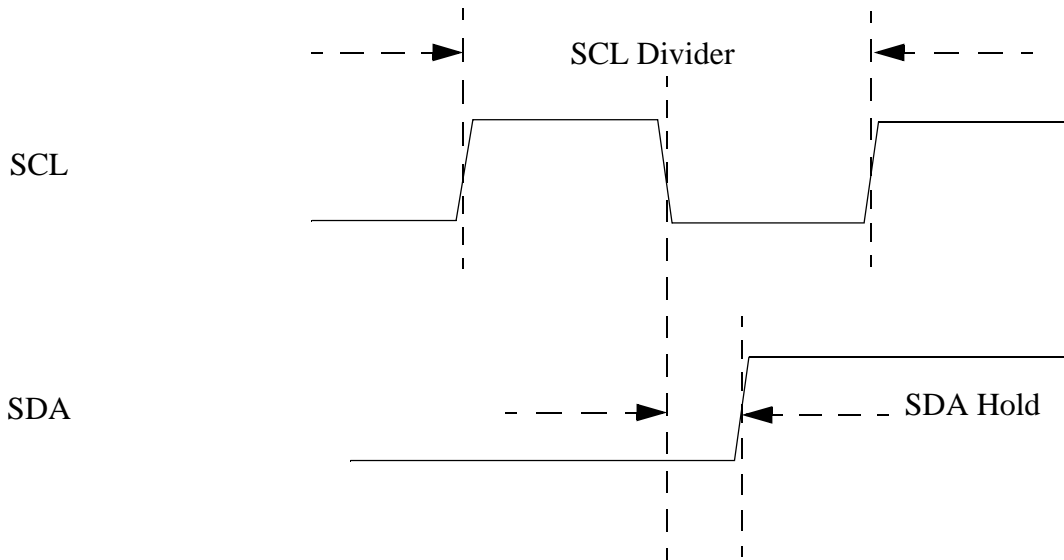


Figure 450. SDA Hold Time

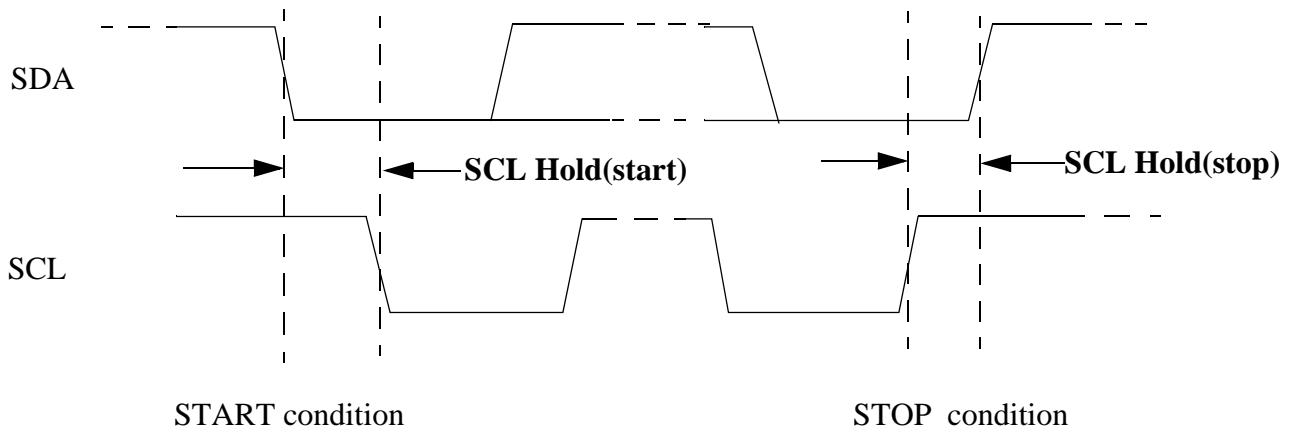


Figure 451. SCL Divider and SDA Hold

The equation used to generate the divider values from the IBFD bits is:

$$\text{SCL Divider} = \text{MUL} \times \{2 \times (\text{scl2tap} + [(\text{SCL_Tap} - 1) \times \text{tap2tap}] + 2)\} \quad \text{Eqn. 1}$$

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 3-4. The equation used to generate the SDA Hold value from the IBFD bits is:

$$\text{SDA Hold} = \text{MUL} \times \{ \text{scl2tap} + [(\text{SDA_Tap} - 1) \times \text{tap2tap}] + 3 \} \quad \text{Eqn. 2}$$

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

$$\text{SCL Hold(start)} = \text{MUL} \times [\text{scl2start} + (\text{SCL_Tap} - 1) \times \text{tap2tap}] \quad \text{Eqn. 3}$$

$$\text{SCL Hold(stop)} = \text{MUL} \times [\text{scl2stop} + (\text{SCL_Tap} - 1) \times \text{tap2tap}] \quad \text{Eqn. 4}$$

Table 458. I²C Divider and Hold Values

	IBC7-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
	24	288	49	142	145
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
30	640	65	318	321	
31	768	65	382	385	
32	896	129	446	449	
33	1024	129	510	513	
34	1152	193	574	577	
35	1280	193	638	641	
36	1536	257	766	769	
37	1920	257	958	961	
38	1280	129	638	641	
39	1536	129	766	769	
3A	1792	257	894	897	
3B	2048	257	1022	1025	
3C	2304	385	1150	1153	
3D	2560	385	1278	1281	
3E	3072	513	1534	1537	
3F	3840	513	1918	1921	

Table 458. I²C Divider and Hold Values (continued)

	IBC7-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
	4A	72	18	28	38
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	28	156	162
	61	384	28	188	194
	62	448	32	220	226
	63	512	32	252	258
	64	576	36	284	290
	65	640	36	316	322
	66	768	40	380	386
	67	960	40	476	482
	68	640	28	316	322
	69	768	28	380	386
	6A	896	36	444	450
	6B	1024	36	508	514
	6C	1152	44	572	578
	6D	1280	44	636	642
	6E	1536	52	764	770
	6F	1920	52	956	962
70	1280	36	636	642	
71	1536	36	764	770	
72	1792	52	892	898	
73	2048	52	1020	1026	
74	2304	68	1148	1154	
75	2560	68	1276	1282	
76	3072	84	1532	1538	
77	3840	84	1916	1922	
78	2560	36	1276	1282	
79	3072	36	1532	1538	
7A	3584	68	1788	1794	
7B	4096	68	2044	2050	
7C	4608	100	2300	2306	
7D	5120	100	2556	2562	
7E	6144	132	3068	3074	
7F	7680	132	3836	3842	

Table 458. I²C Divider and Hold Values (continued)

	IBC7-0 (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL = 4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
30	2560	260	1272	1284	
B1	3072	260	1528	1540	
B2	3584	516	1784	1796	
B3	4096	516	2040	2052	
B4	4608	772	2296	2308	
B5	5120	772	2552	2564	
B6	6144	1028	3064	3076	
B7	7680	1028	3832	3844	
B8	5120	516	2552	2564	
B9	6144	516	3064	3076	
BA	7168	1028	3576	3588	
BB	8192	1028	4088	4100	
BC	9216	1540	4600	4612	
BD	10240	1540	5112	5124	
BE	12288	2052	6136	6148	
BF	15360	2052	7672	7684	

35.3.3.3 I²C Control Register

Offset 0x0002

Access: Supervisor/Test

	0	1	2	3	4	5	6	7
R	MDIS	IBIE	MS/SL	Tx/Rx	NOACK	0	0	IBDOZE
W						RSTA		
Reset	1	0	0	0	0	0	0	0

Figure 452. I²C Bus Control Register (IBCR)
Table 459. IBCR Field Descriptions

Field	Description
7 MDIS	Module disable. This bit controls the software reset of the entire I ² C Bus module. 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed 0 The I ² C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect Note: If the I ² C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I ² C Bus module losing arbitration, after which, bus operation would return to normal.
6 IBIE	I-Bus Interrupt Enable. 1 Interrupts from the I ² C Bus module are enabled. An I ² C Bus interrupt occurs provided the IBIF bit in the status register is also set. 0 Interrupts from the I ² C Bus module are disabled. Note that this does not clear any currently pending interrupt condition
5 MS/SL	Master/Slave mode select. Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. MS/SL is cleared without generating a STOP signal when the master loses arbitration. 1 Master Mode 0 Slave Mode
4 Tx/Rx	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 1 Transmit 0 Receive
3 NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I ² C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. Note that values written to this bit are only used when the I ² C Bus is a receiver, not a transmitter. 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1) 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data
2 RSTA	Repeat Start. Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 1 Generate repeat start cycle 0 No effect

Table 459. IBCR Field Descriptions (continued)

Field	Description
1	Reserved. A read will return 0; should be written as 0.
0 IBSDOZE	<p>I-Bus Interface Stop in DOZE mode.</p> <p>1 Halt I²C Bus module clock generation (if DOZE mode signal asserted) 0 I²C Bus module clock operates normally</p> <p>Note: If the IBSDOZE mode is SET, the I²C module will enter DOZE mode when the DOZE signal is asserted, if there are no current transactions on the bus. The I²C module would then signal to the system that the clock can be shut down.</p> <p>Note: If it were the case that the IBDOZE bit was cleared when the DOZE signal was asserted, the I²C Bus module clock would remain alive, and any current transactions would continue as normal.</p>

35.3.3.4 I²C Status Register

Offset 0x0003

Access: Read-only Supervisor/Test¹

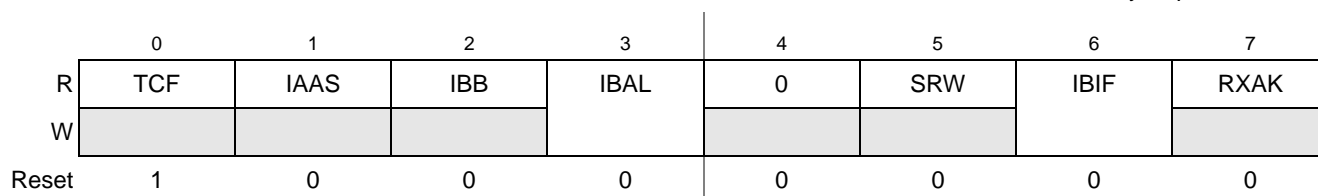


Figure 453. I²C Bus Status Register (IBSR)

¹ With the exception of IBIF and IBAL, which are software clearable.

Table 460. IBSR Field Descriptions

Field	Description
7 TCF	<p>Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the I²C module or from the I²C module.</p> <p>1 Transfer complete 0 Transfer in progress</p>
6 IAAS	<p>Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set its Tx/Rx mode accordingly. Writing to the I-Bus Control Register clears this bit.</p> <p>1 Addressed as a slave 0 Not addressed</p>
5 IBB	<p>Bus busy. This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state.</p> <p>1 Bus is busy 0 Bus is Idle</p>

Table 460. IBSR Field Descriptions (continued)

Field	Description
4 IBAL	Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: <ul style="list-style-type: none"> • SDA is sampled low when the master drives a high during an address or data transmit cycle. • SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
3	Reserved for future use. A read will return 0; should be written as 0.
2 SRW	Slave Read/Write. When IAAS is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By programming this bit, the CPU can select slave transmit/receive mode according to the command of the master. 1 Slave transmit, master reading from slave 0 Slave receive, master writing to slave
1 IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: <ul style="list-style-type: none"> • Arbitration lost (IBAL bit set) • Byte transfer complete (TCF bit set) • Addressed as slave (IAAS bit set) • NoAck from Slave (MS & Tx bits set) • I²C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit.
0 RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. 1 No acknowledge received 0 Acknowledge received

35.3.3.5 I²C Data I/O Register


Figure 454. I²C Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred. Note that the Tx/Rx bit in the IBCR must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I²C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the I²C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I²C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of $\overline{MS}/\overline{SL}$ is used for the address transfer and should comprise the calling address (in position D7–D1) concatenated with the required R/ \overline{W} bit (in position D0).

35.3.3.6 I²C Interrupt Config Register

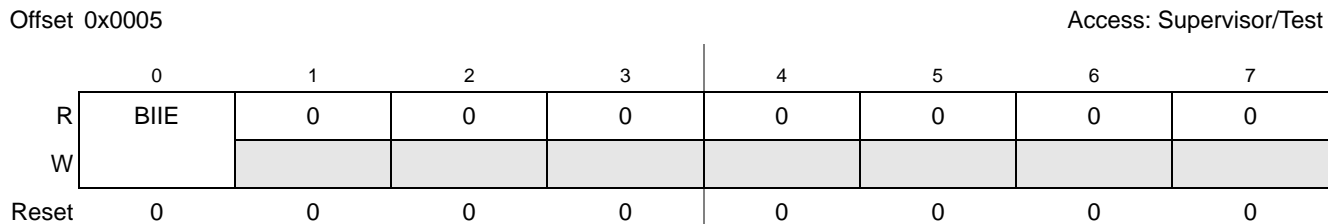


Figure 455. I²C Bus Interrupt Config Register (IBIC)

Table 461. IBIC Field Descriptions

Field	Description
7 BIIE	Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I ² C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I ² C bus. 1 Bus Idle Interrupts enabled 0 Bus Idle Interrupts disabled
6–0	Reserved for future use. A read will return 0; should be written as 0.

35.4 Functional Description

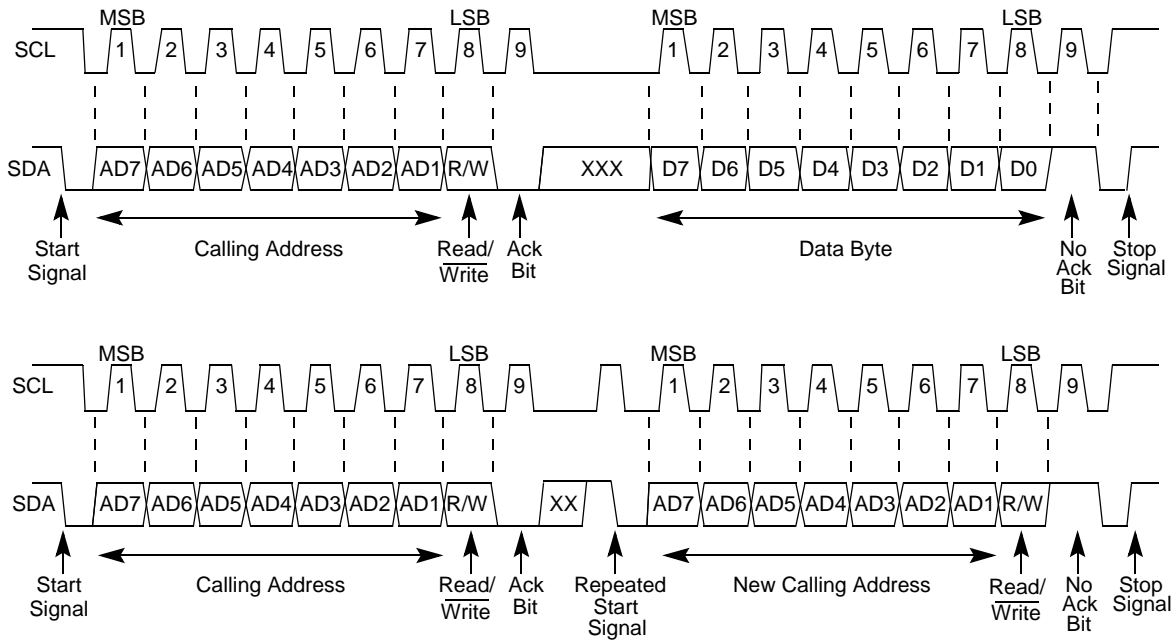
35.4.1 General

This section provides a complete functional description of the Inter-Integrated Circuit (I²C).

35.4.2 I-Bus Protocol

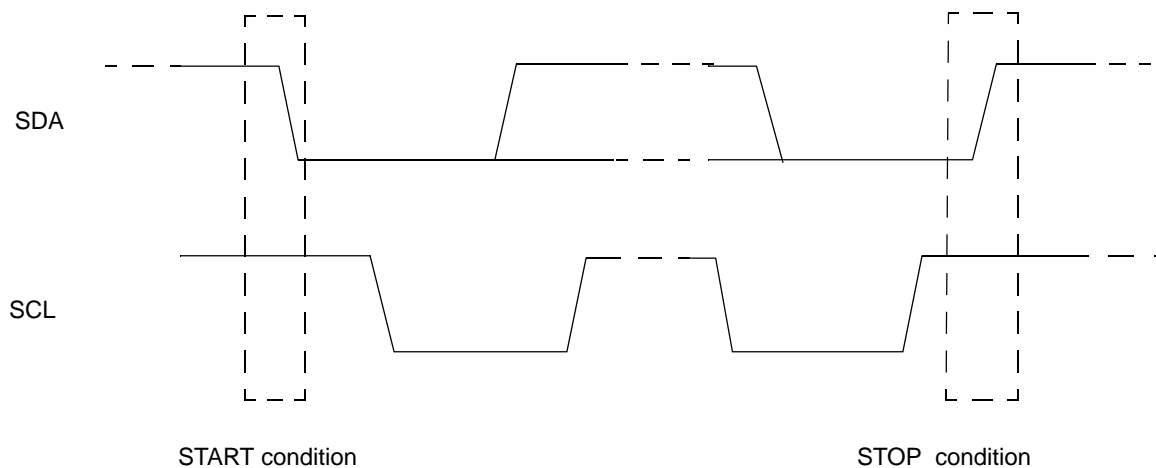
The I²C Bus system uses a Serial Data line (SDA) and a Serial Clock Line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logical AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in [Figure 456](#).


Figure 456. I²C Bus Transmission Signals

35.4.2.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in [Figure 456](#), a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.


Figure 457. Start and Stop conditions

35.4.2.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer - the slave transmits data to the master

0 = Write transfer - the master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see [Figure 456](#)).

No two slaves in the system may have the same address. If the I²C Bus is master, it must not transmit an address that is equal to its own slave address. The I²C Bus cannot be master and slave at the same time. However, if arbitration is lost during an address cycle the I²C Bus will revert to slave mode and operate correctly, even if it is being addressed by another master.

35.4.2.3 Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in [Figure 456](#). There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte must be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. Therefore, one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate a STOP or START signal.

35.4.2.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical "1" (see [Figure 456](#)).

The master can generate a STOP even if the slave has generated an acknowledge, at which point the slave must release the bus.

35.4.2.5 Repeated START Signal

As shown in [Figure 456](#), a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

35.4.2.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

35.4.2.7 Clock Synchronization

Since wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see [Figure 458](#)). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

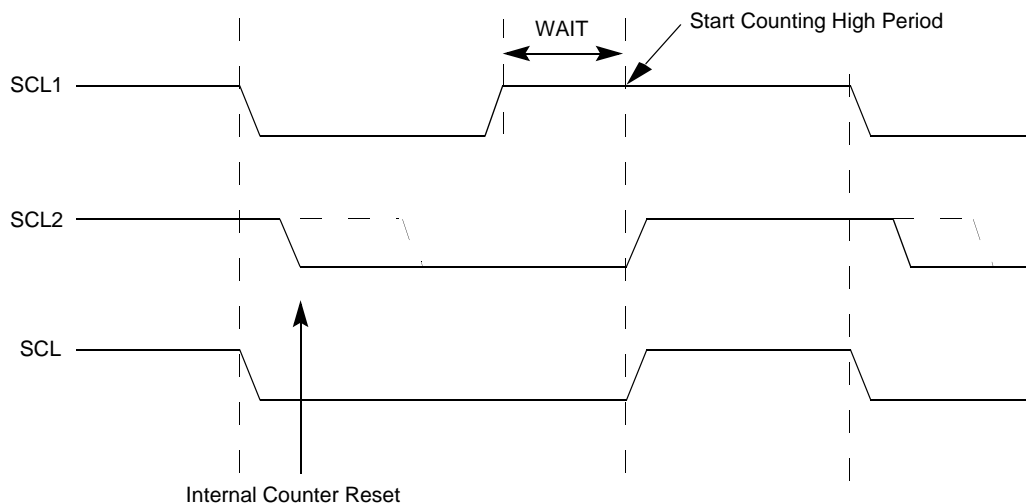


Figure 458. I²C Bus Clock Synchronization

35.4.2.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

35.4.2.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

35.4.3 Interrupts

35.4.3.1 General

The I2C uses only one interrupt vector.

Table 462. Interrupt Summary

Interrupt	Offset	Vector	Priority	Source	Description
I ² C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the deassertion of IBB can also cause an interrupt, indicating that the bus is idle.

35.4.3.2 Interrupt Description

There are five types of internal interrupts in the I²C. The interrupt service routine can determine the interrupt type by reading the Status Register.

I²C Interrupt can be generated on

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I²C interrupt is enabled by the IBIE bit in the I²C Control Register. It must be cleared by writing ‘1’ to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the BIIE bit in the IBIC register.

35.5 Initialization/Application Information

35.5.1 I²C Programming Examples

35.5.1.1 Initialization Sequence

Reset will put the I²C Bus Control Register to its default state. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the Frequency Divider Register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the I²C Bus Address Register (IBAD) to define its slave address.
3. Clear the IBDIS bit of the I²C Bus Control Register (IBCR) to enable the I²C interface system.
4. Modify the bits of the I²C Bus Control Register (IBCR) to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not. Optionally also modify the bits of the I²C Bus Interrupt Config Register (IBIC) to further refine the interrupt behaviour.

35.5.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I²C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I²C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (bit 5, IBSR ==1)// wait in loop for IBB flag to clear
bit4 and bit 5, IBCR = 1// set transmit and master mode, i.e. generate start condition
IBDR = calling_address// send the calling address to the data register
while (bit 5, IBSR ==0)// wait in loop for IBB flag to be set
```

35.5.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I²C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing 1 (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress by reading the IBDR data register in receive mode or writing the IBDR in transmit mode. The TCF bit should not be used as a data transfer

complete flag as the flag timing is dependent on a number of factors including the I²C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag

Software may service the I²C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1) the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IAAS=0) the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example software sequence for 'master transmitter' in the interrupt routine.

```
clear bit 1, IBSR// Clear the IBIF flag
if (bit 5, IBCR ==0)
slave_mode()// run slave mode routine
if (bit 4, IBCR ==0)
receive_mode()// run receive_mode routine
if (bit 0, IBSR == 1)// if NO ACK
    end();// end transmission
else
IBDR = data_to_transmit// transmit next byte of data
```

35.5.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
if (tx_count == 0) or// check to see if all data bytes have been transmitted
    (bit 0, IBSR == 1) { // or if no ACK generated
        clear bit 5, IBCR// generate stop condition
    }
else {
IBDR = data_to_transmit// write byte of data to DATA register
    tx_count --// decrement counter
} // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --// decrease the rx counter
if (rx_count ==1)// 2nd last byte to be read ?
    bit 3, IBCR = 1// disable ACK
if (rx_count == 0)// last byte to be read ?
    bit 1, IBCR = 0// generate stop signal
else
```

```
data_received = IBDR// read RX data and store
```

35.5.1.5 Generation of Repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1// generate another start ( restart)
IBDR == calling_address// transmit the calling address
```

35.5.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

35.5.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

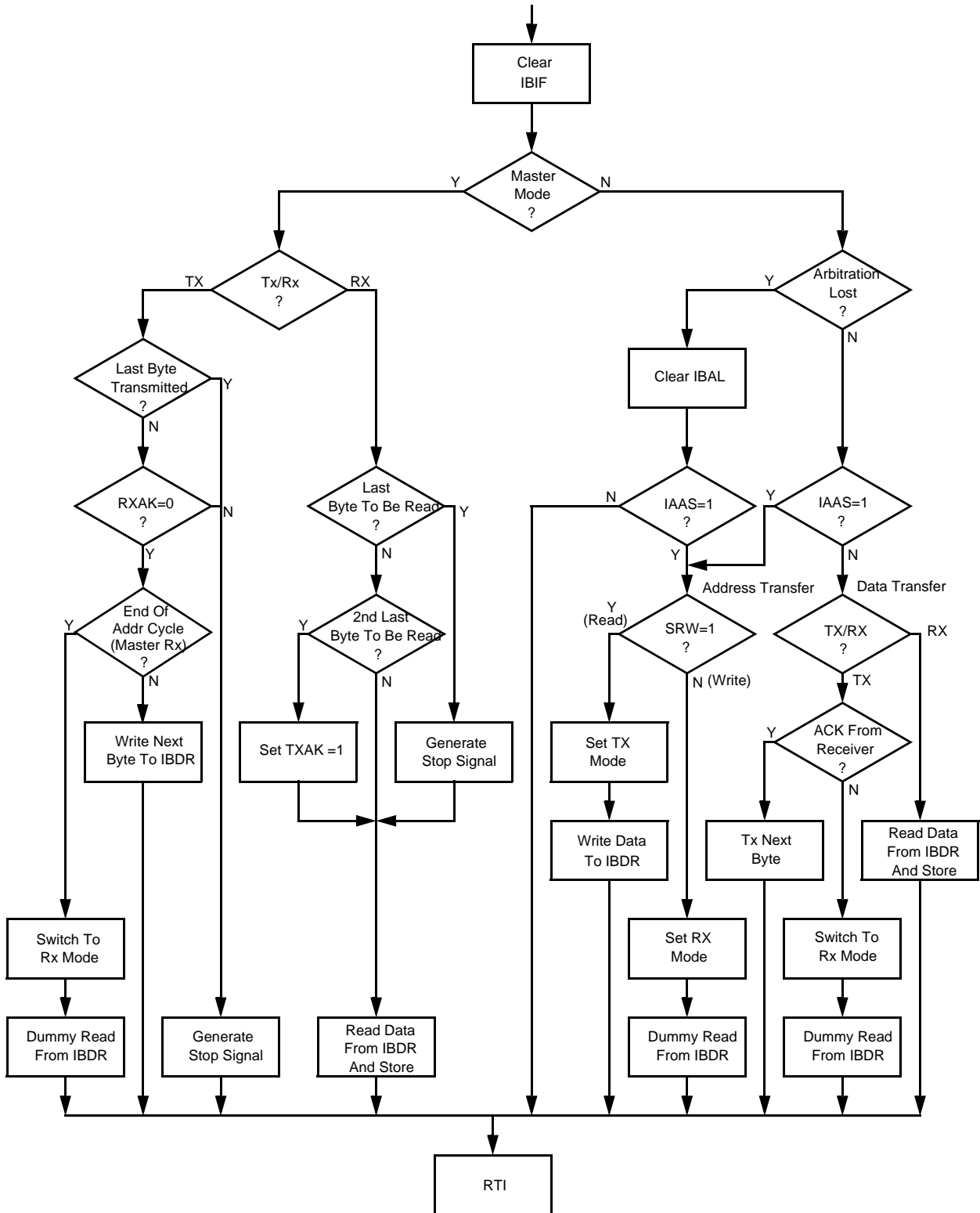


Figure 459. Flow-Chart of Typical I²C Interrupt Routine

Chapter 36

Fast Ethernet Controller (FEC)

This chapter provides a feature-set overview, a functional block diagram, and transceiver connection information for the 10- and 100-Mbps media independent interfaces (MII). Detailed functional descriptions and application/initialization information are included.

36.1 Overview

The fast Ethernet controller (FEC) is designed to support both 10- and 100-Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports three different standard physical interfaces (MAC-PHY) for connection to an external Ethernet transceiver. The FEC is implemented with a combination of hardware and microcode. [Figure 460](#) is a block diagram of the FEC.

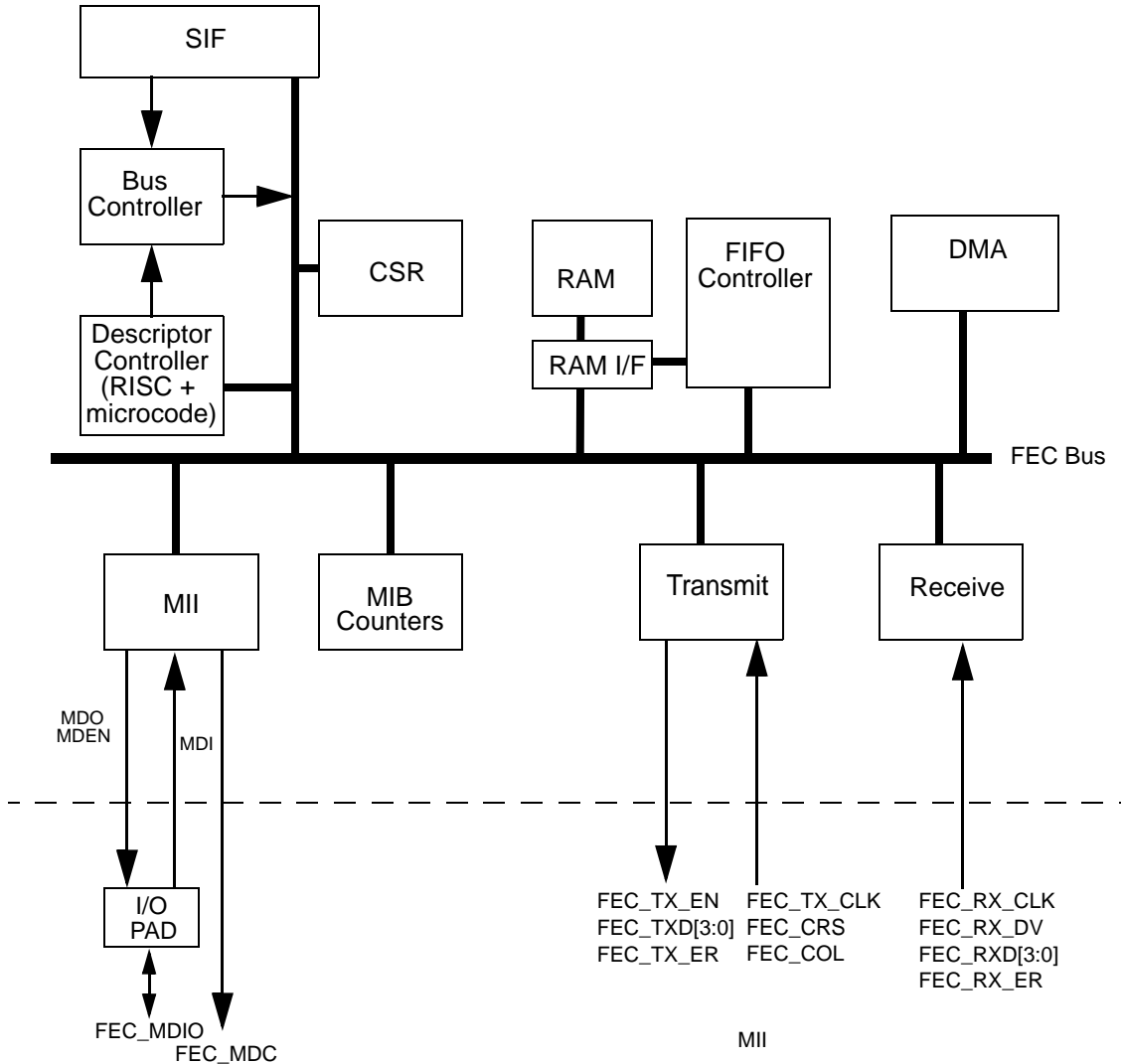


Figure 460. FEC Block Diagram

The FEC submodules shown in [Figure 460](#) are described as follows:

- The descriptor controller is a RISC-based controller that provides the following functions in the FEC:
 - Initialization (those internal registers not initialized by the user or hardware)
 - High-level control of the DMA channels (initiating DMA transfers)
 - Interpreting buffer descriptors
 - Address recognition for receive frames
 - Random number generation for transmit collision backoff timer

NOTE

This DMA engine is for the transfer of FEC data only and is not related to the system DMA controller.

- The RAM is the focal point of all data flow in the FEC and is divided into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to and from the DMA block, from and to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO to the transmit block, and receive data flows from the receive block to the receive FIFO.
- The user controls the FEC by writing, through the slave interface (SIF) submodule, to control registers located in each block. The control and status register (CSR) block provides global control (for example, Ethernet reset and enable) and interrupt handling registers.
- The MII block provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the management data clock and management data input/output lines of the MII interface (FEC_MDC and FEC_MDIO, respectively).
- The DMA block provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to run independently.
- The transmit and receive blocks provide the Ethernet MAC functionality (with some assistance from microcode).
- The message information block (MIB) maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet statistics group and some of the IEEE 802.3 counters. See [Section 36.3.3, “Message Information Block \(MIB\) Counters Memory Map”](#), for more information.

36.1.1 Features

The FEC incorporates the following features:

- Support for three different Ethernet physical interfaces:
 - 10-Mbps and 100-Mbps IEEE Std. 802.3™ MII
- IEEE 802.3 full-duplex flow control and half duplex flow
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200Mbps throughput) with a minimum system clock rate of 50 MHz
- Support for half-duplex operation (100Mbps throughput) with a minimum system clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
 - Frames with broadcast address can be always accepted or always rejected
 - Exact match for single 48-bit individual (unicast) address
 - Hash (64-bit hash) check of individual (unicast) addresses
 - Hash (64-bit hash) check of group (multicast) addresses

— Promiscuous mode

36.2 Modes of Operation

The primary operational modes are described in this section.

36.2.1 Full- and Half-Duplex Operation

Full-duplex mode is intended for use on point to point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters. Selection of the duplex mode is controlled by the TCR[FDEN] bit. When configured for full-duplex mode, flow control can be enabled, which is effected by the TCR[RFC_PAUSE], TCR[TFC_PAUSE], and RCR[FCE] bits. See [Section 36.4.4, “Full-Duplex Flow Control”](#), for more details.

36.2.2 Interface Options

The following interface options are supported. A detailed discussion of the interface configurations is provided in [Section 36.4.1, “Network Interface Options”](#).

36.2.2.1 10-Mbps and 100-Mbps Media Independent Interface (MII)

The MII is defined by the IEEE 802.3 standard for 10/100-Mbps operation. The MAC-PHY interface can be configured to operate in MII mode by asserting RCR[MII_MODE].

The speed of operation is determined by the FEC_TX_CLK and FEC_RX_CLK signals which are driven by the external transceiver. The transceiver either autonegotiates the speed or control by software via the serial management interface (FEC_MDC/FEC_MDIO) signals to the transceiver. See the descriptions in [Section 36.3.4.6, “MII Management Frame Register \(MMFR\)”](#) and [Section 36.3.4.7, “MII Speed Control Register \(MSCR\)”](#) respectively, as well as MII documentation for a description of how to read and write registers in the transceiver via this interface.

36.2.3 Address Recognition Options

The address options supported are promiscuous, broadcast reject, individual address (hash or exact match), and multicast hash match. Address recognition options are discussed in detail in [Section 36.4.3.2, “Ethernet Address Recognition”](#).

36.2.4 Internal Loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in [Section 36.4.5, “Internal and External Loopback”](#).

36.3 Memory Map and Register Definition

This section includes the FEC memory map and detailed descriptions of all the registers, followed by a description of the buffers.

The FEC is programmed by a combination of control and status registers (CSRs) and buffer descriptors. The CSRs are used for mode control and to extract global status information. The descriptors are used to pass data buffers and related buffer information between the hardware and software.

36.3.1 Top Level Module Memory Map

The FEC implementation requires a 1 KB memory space. This space is divided into 2 sections of 512 bytes each. The first is used for control/status registers, and the second contains event/statistics counters held in the MIB block.

[Table 463](#) defines the top level memory map. For the base address of a particular module instantiation, see the system memory map.

Table 463. Module Memory Map

Base Address Offset	Function
0x0000–01FF	Control/Status Registers
0x0200–03FF	MIB Block Counters

36.3.2 Detailed Memory Map (Control/Status Registers)

[Table 464](#) shows the FEC register memory map. For the base address of a particular module instantiation, see the system memory map.

Table 464. FEC Registers Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
General Registers				
0x0004 (EIR)	Ethernet interrupt event register	R/W	0x0000_0000	36.3.4.1/832
0x0008 (EIMR)	Ethernet interrupt mask register	R/W	0x0000_0000	36.3.4.2/834
0x0010 (RDAR)	Receive descriptor active register	R/W	0x0000_0000	36.3.4.3/835
0x0014 (TDAR)	Transmit descriptor active register	R/W	0x0000_0000	36.3.4.4/836
0x0024 (ECR)	Ethernet control register	R/W	0xF000_0004	36.3.4.5/837
0x0040 (MMFR)	MII management frame register	R/W	Unaffected by reset	36.3.4.6/838
0x0044 (MSCR)	MII speed control register	R/W	0x0000_0000	36.3.4.7/839
0x0064 (MIBC)	MIB control register	R/W	0xC000_0000	36.3.4.8/841
0x0084 (RCR)	Receive control register	R/W	0x05EE_0001	36.3.4.9/841
0x00C4 (TCR)	Transmit control register	R/W	0x0000_0000	36.3.4.10/843
0x00E4 (PALR)	Physical address low register	R/W	Unaffected by reset	36.3.4.11/844
0x00E8 (PAUR)	Physical address upper register	Mixed	0xuuuu_8808	36.3.4.12/845
0x00EC (OPDR)	Opcode and pause duration register	Mixed	0x0001_uuuu	36.3.4.13/845

Table 464. FEC Registers Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0118 (IAUR)	Descriptor individual address upper register	R/W	Unaffected by reset	36.3.4.14/846
0x011C (IALR)	Descriptor individual address lower register	R/W	Unaffected by reset	36.3.4.15/847
0x0120 (GAUR)	Descriptor group address upper register	R/W	Unaffected by reset	36.3.4.16/847
0x0124 (GALR)	Descriptor group address lower register	R/W	Unaffected by reset	36.3.4.17/848
0x0144 (TFWR)	Transmit FIFO watermark register	R/W	0x0000_0000	36.3.4.18/849
0x014C (FRBR)	FIFO receive bound register	R/O	0x0000_0600	36.3.4.19/849
0x0150 (FRSR)	FIFO receive FIFO start registers	R/W	0x0000_0500	36.3.4.20/850
0x0180 (ERDSR)	Receive buffer descriptor ring start register	R/W	Unaffected by reset	36.3.4.21/851
0x0184 (ETDSR)	Transmit buffer descriptor ring start register	R/W	Unaffected by reset	36.3.4.22/852
0x0188 (EMRBR)	Maximum receive buffer size register	R/W	Unaffected by reset	36.3.4.23/852

36.3.3 Message Information Block (MIB) Counters Memory Map

Table 465 shows the MIB counters memory map, which defines the locations in the MIB RAM space where hardware maintained counters reside. It is the responsibility of software to poll the counters often enough to ensure that rollover is detected. For example, on a 100 Mbps channel an octets counter could roll over every 5.7 minutes.

These counters fall in the 0x0200-0x03FF address offset range, and are divided into RMON counters and IEEE counters, as follows:

- RMON counters are included which cover the Ethernet statistics counters defined in RFC 1757. In addition to the counters defined in the Ethernet statistics group, a counter is included to count truncated frames as the FEC only supports frame lengths up to 2047 bytes. The RMON counters are implemented independently for transmit and receive to insure accurate network statistics when operating in full-duplex mode.
- IEEE counters are included which support the mandatory and recommended counter packages defined in section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The IEEE basic package objects are supported by the FEC but do not require counters in the MIB block. In addition, some of the recommended package objects which are supported do not require MIB counters. Counters for transmit and receive full-duplex flow control frames are included as well.

Table 465. MIB Counters Memory Map

Base Address Offset	Mnemonic	Description
0x0200	RMON_T_DROP	Count of frames not counted correctly
0x0204	RMON_T_PACKETS	RMON Tx packet count

Table 465. MIB Counters Memory Map (continued)

Base Address Offset	Mnemonic	Description
0x0208	RMON_T_BC_PKT	RMON Tx broadcast packets
0x020C	RMON_T_MC_PKT	RMON Tx multicast packets
0x0210	RMON_T_CRC_ALIGN	RMON Tx packets w CRC/Align error
0x0214	RMON_T_UNDERSIZE	RMON Tx packets < 64 bytes, good crc
0x0218	RMON_T_OVERSIZE	RMON Tx packets > MAX_FL bytes, good crc
0x021C	RMON_T_FRAG	RMON Tx packets < 64 bytes, bad crc
0x0220	RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad crc
0x0224	RMON_T_COL	RMON Tx collision count
0x0228	RMON_T_P64	RMON Tx 64 byte packets
0x022C	RMON_T_P65TO127	RMON Tx 65 to 127 byte packets
0x0230	RMON_T_P128TO255	RMON Tx 128 to 255 byte packets
0x0234	RMON_T_P256TO511	RMON Tx 256 to 511 byte packets
0x0238	RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets
0x023C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets
0x0240	RMON_T_P_GTE2048	RMON Tx packets w > 2048 bytes
0x0244	RMON_T_OCTETS	RMON Tx octets
0x0248	IEEE_T_DROP	Count of frames not counted correctly
0x024C	IEEE_T_FRAME_OK	Frames transmitted OK
0x0250	IEEE_T_1COL	Frames transmitted with single collision
0x0254	IEEE_T_MCOL	Frames transmitted with multiple collisions
0x0258	IEEE_T_DEF	Frames transmitted after deferral delay
0x025C	IEEE_T_LCOL	Frames transmitted with late collision
0x0260	IEEE_T_EXCOL	Frames transmitted with excessive collisions
0x0264	IEEE_T_MACERR	Frames transmitted with Tx FIFO underrun
0x0268	IEEE_T_CSERR	Frames transmitted with carrier sense error
0x026C	IEEE_T_SQE	Frames transmitted with SQE error
0x0270	IEEE_T_FDXFC	Flow control pause frames transmitted
0x0274	IEEE_T_OCTETS_OK	Octet count for frames transmitted w/o error
0x0284	RMON_R_PACKETS	RMON Rx packet count
0x0288	RMON_R_BC_PKT	RMON Rx broadcast packets
0x028C	RMON_R_MC_PKT	RMON Rx multicast packets
0x0290	RMON_R_CRC_ALIGN	RMON Rx packets w CRC/Align error

Table 465. MIB Counters Memory Map (continued)

Base Address Offset	Mnemonic	Description
0x0294	RMON_R_UNDERSIZE	RMON Rx packets < 64 bytes, good crc
0x0298	RMON_R_OVERSIZE	RMON Rx packets > MAX_FL bytes, good crc
0x029C	RMON_R_FRAG	RMON Rx packets < 64 bytes, bad crc
0x02A0	RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad crc
0x02A4	RMON_R_RESVD_0	—
0x02A8	RMON_R_P64	RMON Rx 64 byte packets
0x02AC	RMON_R_P65TO127	RMON Rx 65 to 127 byte packets
0x02B0	RMON_R_P128TO255	RMON Rx 128 to 255 byte packets
0x02B4	RMON_R_P256TO511	RMON Rx 256 to 511 byte packets
0x02B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets
0x02BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets
0x02C0	RMON_R_P_GTE2048	RMON Rx packets w > 2048 bytes
0x02C4	RMON_R_OCTETS	RMON Rx octets
0x02C8	IEEE_R_DROP	Count of frames not counted correctly
0x02CC	IEEE_R_FRAME_OK	Frames received OK
0x02D0	IEEE_R_CRC	Frames received with CRC error
0x02D4	IEEE_R_ALIGN	Frames received with alignment error
0x02D8	IEEE_R_MACERR	Receive FIFO overflow count
0x02DC	IEEE_R_FDXFC	Flow control pause frames received
0x02E0	IEEE_R_OCTETS_OK	Octet count for frames received w/o error

36.3.4 Register Descriptions

This section provides detailed descriptions of the FEC’s registers.

36.3.4.1 Ethernet Interrupt Event Register (EIR)

The EIR bit assignments are shown in [Figure 461](#) and described in [Table 466](#). When an event occurs that sets a bit in the EIR, an interrupt is generated if the corresponding bit in the interrupt mask register (EIMR) is also set. The bit in the EIR is cleared if a one is written to that bit position; writing zero has no effect. This register is cleared upon hardware reset.

Interrupts can be divided into three classes as follows:

- Interrupts that occur in normal operation: these include GRA, TXF, TXB, RXF, RXB, and MII.
- Interrupts that result from errors or problems detected in the network or transceiver: these include HBERR, BABR, BABT, LC, and RL.

- Interrupts that result from internal errors are EBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters. The correspondence between interrupts and counters is shown in [Table 467](#). Software can choose to mask off the interrupts since these errors is visible to network management via the MIB counters.

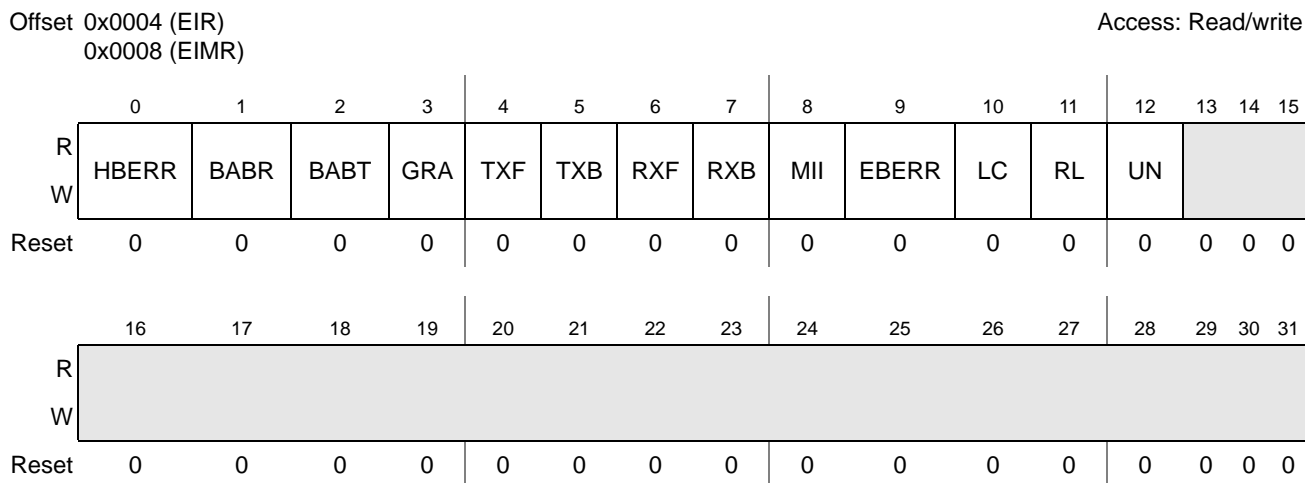


Figure 461. Ethernet Interrupt Event Register (EIR)

Table 466. EIR Field Descriptions

Bits	Name	Description
0	HBERR	Heartbeat error. This interrupt indicates that HBC is set in the TCR register and that the COL input was not asserted within the Heartbeat window following a transmission.
1	BABR	Babbling receive error. This bit indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded RCR[MAX_FL] bytes. This condition is usually caused by a frame that is too long being placed into the transmit data buffer(s). Truncation does not occur.
3	GRA	Graceful stop complete. This interrupt is asserted for one of three reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. 1) A graceful stop, which was initiated by the setting of the TCR[GTS] bit is now complete. 2) A graceful stop, which was initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. 3) A graceful stop, which was initiated by the reception of a valid full-duplex flow control “pause” frame is now complete. See the “Full-Duplex Flow Control” section of the Functional Description chapter.
4	TXF	Transmit frame interrupt. This bit indicates that a frame has been transmitted and that the last corresponding buffer descriptor has been updated.
5	TXB	Transmit buffer interrupt. This bit indicates that a transmit buffer descriptor has been updated.
6	RXF	Receive frame interrupt. This bit indicates that a frame has been received and that the last corresponding buffer descriptor has been updated.

Table 466. EIR Field Descriptions (continued)

Bits	Name	Description
7	RXB	Receive buffer interrupt. This bit indicates that a receive buffer descriptor has been updated that was not the last in the frame.
8	MII	MII interrupt. This bit indicates that the MII has completed the data transfer requested.
9	EBERR	Ethernet bus error. This bit indicates that a system bus error occurred when a DMA transaction was underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs software needs to insure that the FIFO controller and DMA are also soft reset.
10	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded.
11	RL	Collision retry limit. This bit indicates that a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. Can only occur in half-duplex mode.
12	UN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13–31	—	Reserved, read as 0

Table 467. Error Interrupts and Block Counters

Interrupt	Counter(s)
HBERR	IEEE_T_SQE
BABR	RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
BABT	RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
LATE_COL	IEEE_T_LCOL
COL_RETRY_LIM	IEEE_T_EXCOL
XFIFO_UN	IEEE_T_MACERR

36.3.4.2 Ethernet Interrupt Mask Register (EIMR)

The EIMR controls which of the interrupt events flagged in the EIR are allowed to generate actual interrupts. If the corresponding bits in both the EIR and EIMR are set, the interrupt is signaled to the CPU. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit. This register is cleared upon a hardware reset.

The EIMR bit assignments are the same as for the EIR: they are shown in [Figure 461](#) and described in [Table 468](#).

Table 468. EIMR Field Descriptions

Bits	Name	Description
0–12	See Table 466 .	Interrupt mask. Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
13–31	—	Reserved, read as 0

36.3.4.3 Receive Descriptor Active Register (RDAR)

RDAR is a user-writeable command register which indicates that the receive descriptor ring has been updated, and that empty receive buffers have been produced by the driver with the empty bit set.

The RDAR[R_DES_ACTIVE] bit is set whenever the register is written, independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (provided ECR[ETHER_EN] is also set to 1). After the FEC polls a receive descriptor whose empty bit is not set, then the FEC clears the RDAR[R_DES_ACTIVE] bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors have been placed into the receive descriptor ring.

The RDAR is cleared at reset, and when ECR[ETHER_EN] is cleared.

The RDAR bit assignments are shown in [Figure 462](#) and described in [Table 469](#).

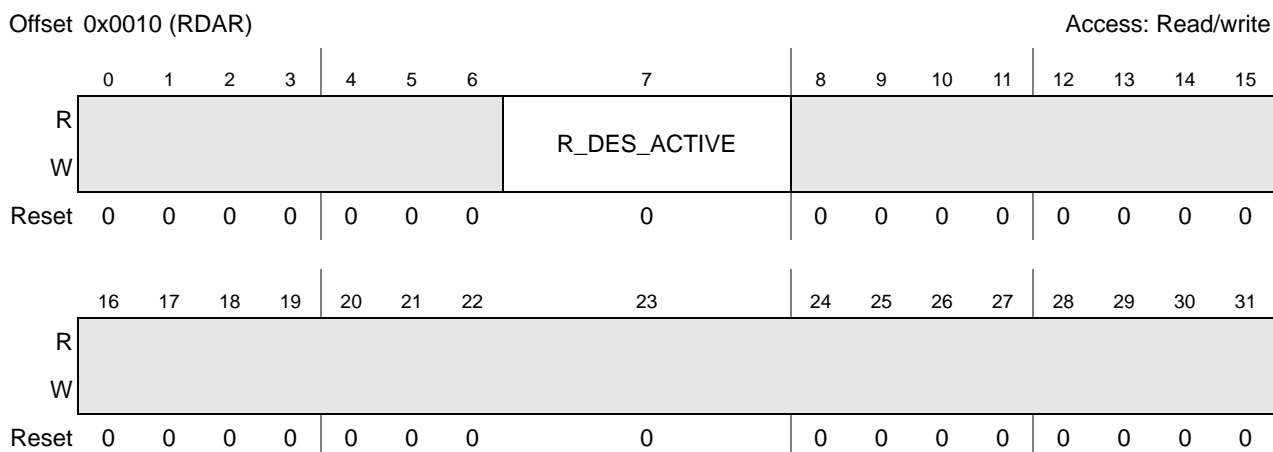

Figure 462. Receive Descriptor Active Register (RDAR)

Table 469. RDAR Field Descriptions

Bits	Name	Description
0–6	—	Reserved, read as 0
7	R_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device when the FEC polls a receive descriptor whose empty bit is not set. Also cleared when ECR[ETHER_EN] is cleared.
8–31	—	Reserved, read as 0

36.3.4.4 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register, written to by the user, to indicate that the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

Whenever the register is written the TDAR[X_DES_ACTIVE] bit is set to 1, independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and process transmit frames (provided ECR[ETHER_EN] is also set to 1). After the FEC polls a transmit descriptor whose ready bit is not set, then the FEC clears the TDAR[X_DES_ACTIVE] bit and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors have been placed into the transmit descriptor ring.

The TDAR is cleared at reset, when ECR[ETHER_EN] is cleared, or when ECR[RESET] is set to 1.

The TDAR bit assignments are shown in [Figure 463](#) and described in [Table 470](#).

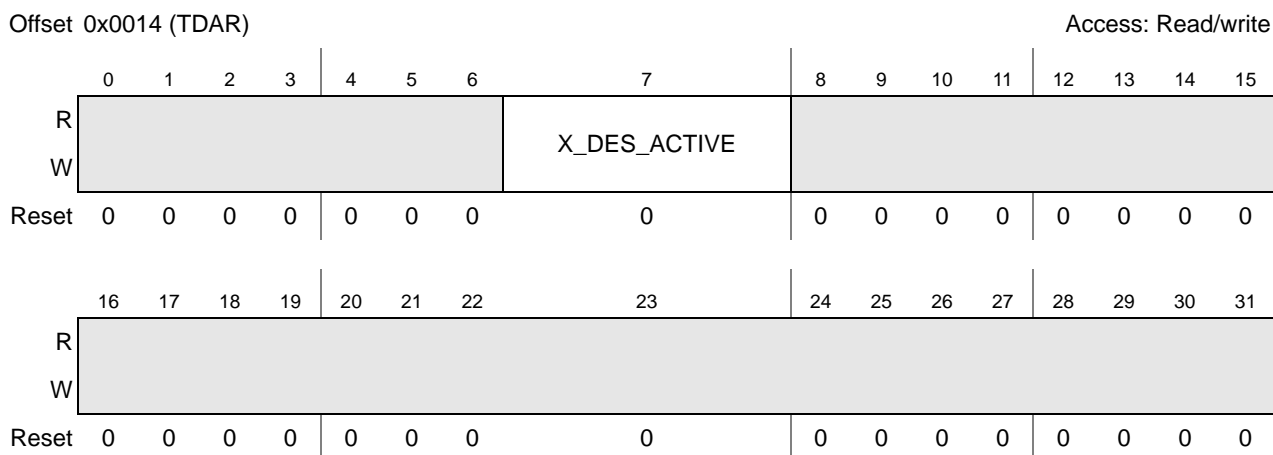


Figure 463. Transmit Descriptor Active Register (TDAR)

Table 470. TDAR Field Descriptions

Bits	Name	Description
0–6	—	Reserved, read as 0

Table 470. TDAR Field Descriptions (continued)

Bits	Name	Description
7	X_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device when the FEC polls a transmit descriptor whose ready bit is not set. Also cleared when ECR[ETHER_EN] is cleared.
8–31	—	Reserved, read as 0

36.3.4.5 Ethernet Control Register (ECR)

The ECR is used to enable/disable the FEC. ECR is a read/write user register, though both fields in this register can also be altered by hardware.

ECR bit assignments are shown in [Figure 464](#) and described in [Table 471](#).

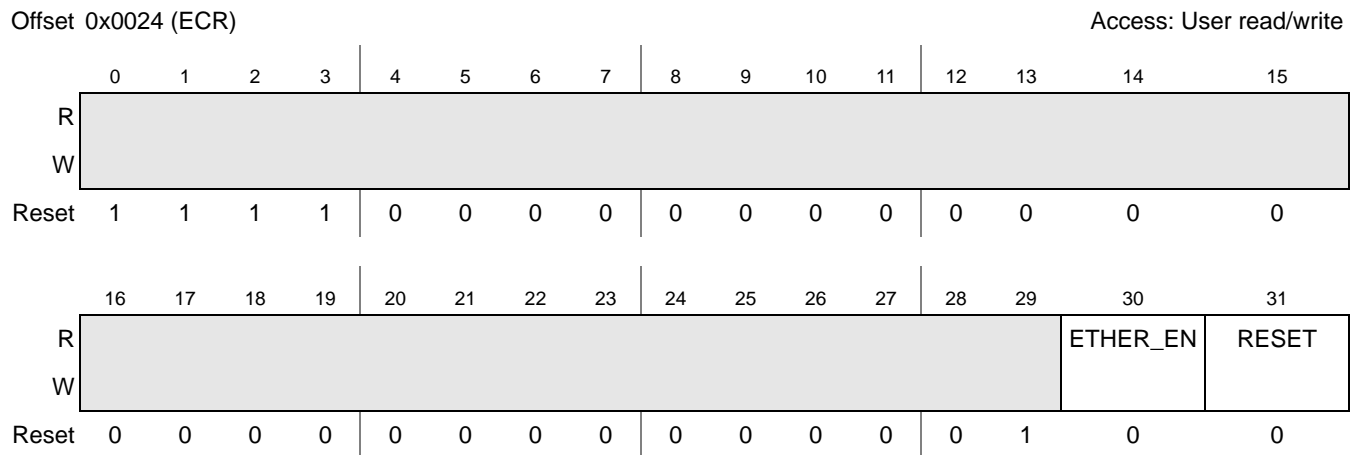


Figure 464. Ethernet Control Register (ECR)

Table 471. ECR Field Descriptions

Bits	Name	Description
0-29	—	Reserved.

Table 471. ECR Field Descriptions (continued)

Bits	Name	Description
30	ETHER_EN	When this bit is set, the FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception is immediately stopped and transmission is stopped after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. The ETHER_EN bit is altered by hardware under the following conditions: <ul style="list-style-type: none"> • ECR[RESET] is set by software, in which case ETHER_EN is cleared • an error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared
31	RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ETHER_EN is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 system clock cycles after RESET is written with a 1.

36.3.4.6 MII Management Frame Register (MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to MMFR causes a management frame to be generated unless the MII-SPEED field of the MSCR has been set to 0, in which case MSCR is set to a non-zero value and an MII frame is generated with the data previously written to MMFR. This allows MMFR and MSCR to be programmed in either order if the MII-SPEED field of MSCR is zero (for further details see [Section 36.3.4.7, “MII Speed Control Register \(MSCR\)”](#)).

The MMFR does not reset to a definite value. MMFR bit assignments are shown in [Figure 465](#) and described in [Table 472](#).

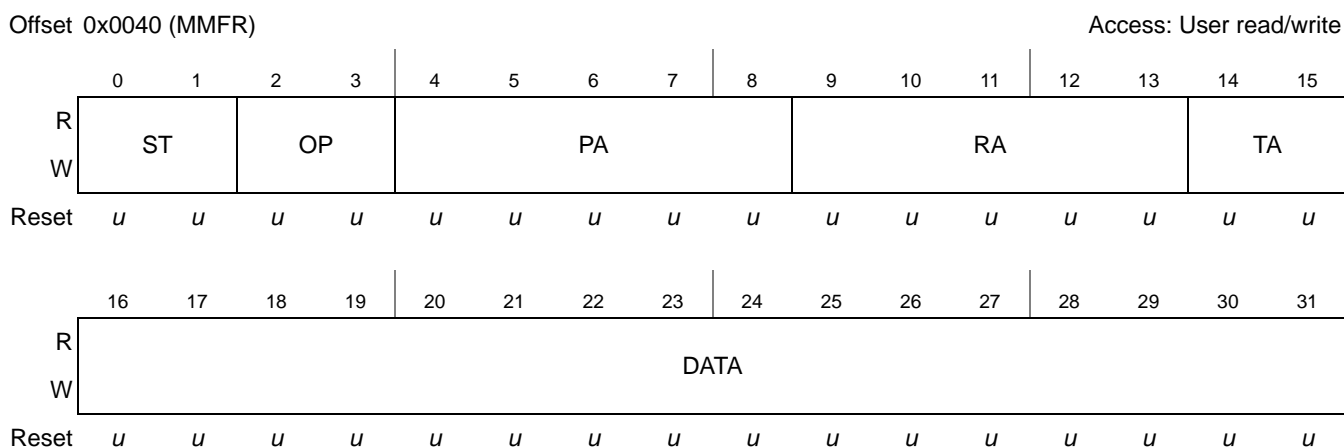


Figure 465. MII Management Frame Register (MMFR)

Table 472. MMFR Field Descriptions

Bits	Name	Description
0–1	ST	Start of frame delimiter. These bits must be programmed to 01 for a valid MII management frame.
2–3	OP	Operation code. This field must be programmed to 10 (read) or 01 (write) to generate a valid MII management frame. A value of 11 produces “read” frame operation while a value of 00 produces “write” frame operation, but these frames is not MII compliant.
4–8	PA	PHY address. This field specifies one of up to 32 attached PHY devices.
9–13	RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
14–15	TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
16–31	DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII Management Interface, the MMFR must be written to by the user. To generate a valid read or write management frame, the ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII management interface write frame (write to a PHY register), the user must write {01 01 PHYAD REGAD 10 DATA} to the bit fields of the MMFR, as shown in [Table 472](#). Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time the contents of the MMFR is altered as the contents are serially shifted and is unpredictable if read by the user. After the write management frame operation has completed, the MII interrupt is generated. At this time the contents of the MMFR matches the original value written.

To generate an MII management interface read frame (read a PHY register) the user must write {01 10 PHYAD REGAD 10 XXXX} to the bit fields of the MMFR shown in [Table 472](#) (the contents of the 4-bit DATA field are arbitrary). Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time the contents of the MMFR is altered as the contents are serially shifted, and is unpredictable if read by the user. After the read management frame operation has completed, the MII interrupt is generated. At this time the contents of the MMFR matches the original value written except for the DATA field whose contents have been replaced by the value read from the PHY register.

If the MMFR is written to while frame generation is in progress, the frame contents is altered. Software uses the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

36.3.4.7 MII Speed Control Register (MSCR)

MSCR bit assignments are shown in [Figure 466](#) and described in [Table 473](#). The MSCR provides control of the frequency of the MII clock (FEC_MDC signal), and allows a preamble drop on the MII management frame.

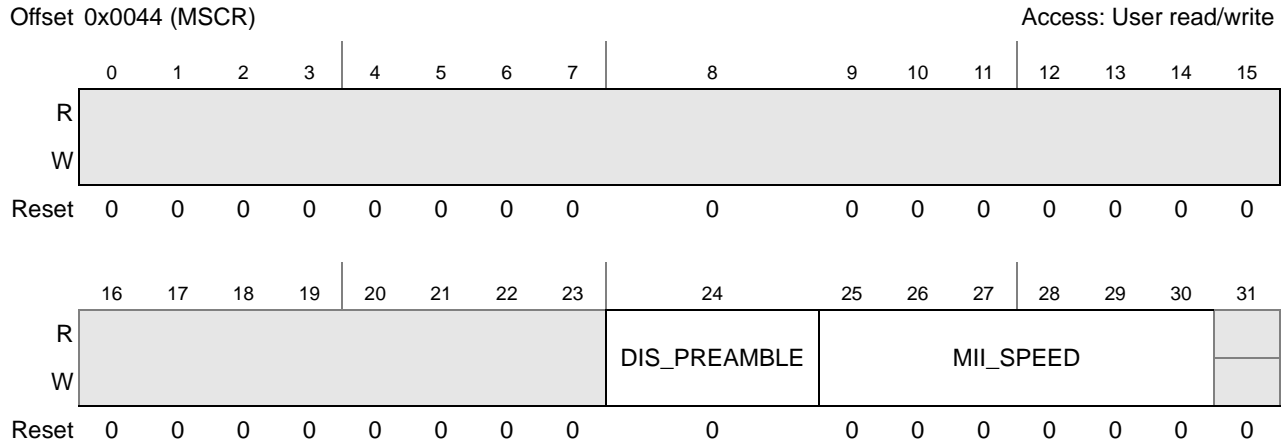


Figure 466. MII Speed Control Register (MSCR)

Table 473. MSCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved, read as 0
24	DIS_PREAMBLE	Asserting this bit causes preamble (0xFFFF_FFFF) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
25–30	MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (FEC_MDC) relative to the system clock. A value of 0 in this field “turns off” the FEC_MDC and leave it in low voltage state. Any non-zero value results in the FEC_MDC frequency of $1/(MII_SPEED*2)$ of the system clock frequency.
31	—	Reserved, read as 0

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value in order to generate a read or write management frame. After the management frame is complete the MSCR can optionally be set to zero to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED is changed during operation (change takes effect following either a rising or falling edge of FEC_MDC).

The FEC_MDC frequency depends on both the system clock frequency and the MII_SPEED register. If the system clock is 25 MHz, programming the MII_SPEED register to 0x0000_0005 results in an FEC_MDC frequency of $25\text{ MHz} * 1/10 = 2.5\text{ MHz}$. A table showing optimum values for MII_SPEED for different system clock frequencies is provided below.

Table 474. Programming Examples for MSCR

System Clock Frequency	MII_SPEED (field in reg)	FEC_MDC frequency
25 MHz	0x5	2.5 MHz
33 MHz	0x7	2.36 MHz
40 MHz	0x8	2.5 MHz

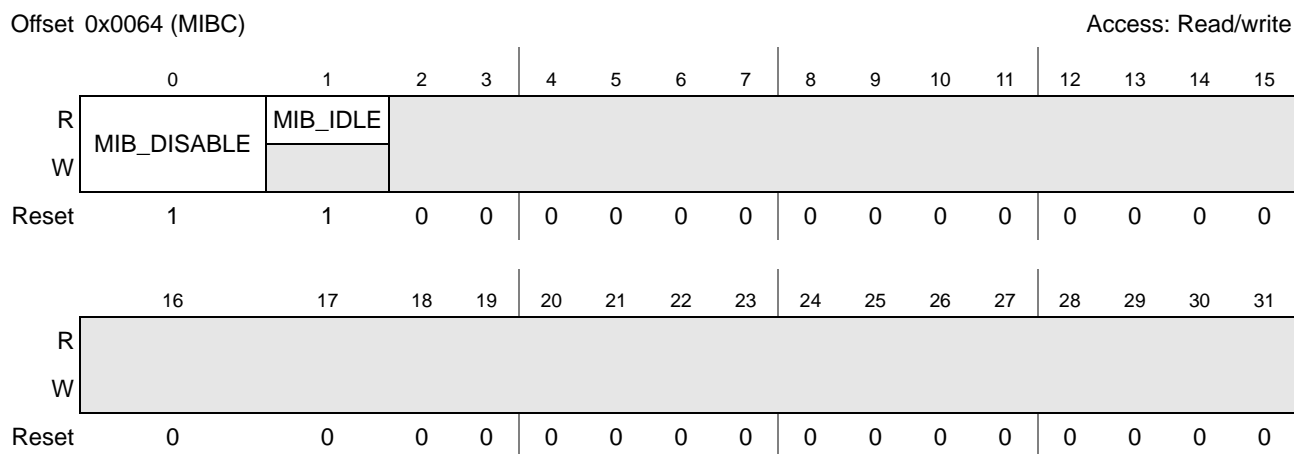
Table 474. Programming Examples for MSCR

System Clock Frequency	MII_SPEED (field in reg)	FEC_MDC frequency
50 MHz	0xA	2.5 MHz
66 MHz	0xD	2.54 MHz

36.3.4.8 MIB Control Register (MIBC)

The MIB control register is a read/write register used to provide control of and to observe the state of the Message Information Block (MIB). This register is accessed by user software if there is a need to disable the MIB operation. For example, in order to clear all MIB counters in RAM the user disables the MIB, then clears all the MIB RAM locations, then enables the MIB. The MIB_DISABLE bit is reset to 1. See [Table 465](#) for the locations of the MIB counters.

MIBC bit assignments are shown in [Figure 467](#) and described in [Table 475](#).


Figure 467. MIB Control Register (MIBC)
Table 475. MIBC Field Descriptions

Bits	Name	Description
0	MIB_DISABLE	A read/write control bit. If set, the MIB logic halts and not update any MIB counters.
1	MB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
2–31	—	Reserved.

36.3.4.9 Receive Control Register (RCR)

RCR bit fields are shown in [Figure 468](#) and described in [Table 476](#). The RCR is programmed by the user, and controls the operational mode of the receive block. It can only be written to when ECR[ETHER_EN] = 0 (that is, during initialization).

Offset 0x0084 (RCR)

Access: Read/write

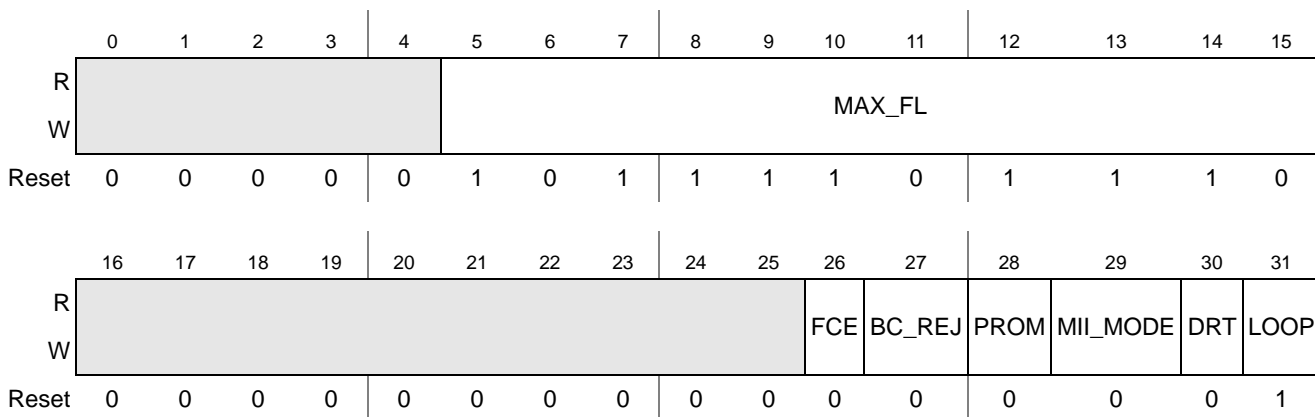


Figure 468. Receive Control Register (RCR)

Table 476. RCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, read as 0
5–15	MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BAPT interrupt to occur. Receive Frames longer than MAX_FL causes the BAPT interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed by the user is 1518 or 1522 (if VLAN Tags are supported).
16–25	—	Reserved, read as 0
26	FCE	Flow control enable. When FCE is set to 1, the receiver detects pause frames. Upon pause frame detection, the transmitter stops transmitting data frames for a given duration.
27	BC_REJ	Broadcast frame reject. When BC_REJ is set to 1, frames with DA (destination address) = 0xFF_FF_FF_FF_FF_FF are rejected unless the PROM bit is set to 1. If both BC_REJ and PROM are set to 1, then frames with broadcast DA is accepted and the M (MISS) bit is set in the receive buffer descriptor.
28	PROM	Promiscuous mode. All frames are accepted regardless of address matching.
29	MII_MODE	Media independent interface mode. Selects external interface mode. 1) Setting this bit to one selects MII mode. This bit controls the interface mode for both transmit and receive blocks.
30	DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full-duplex or to monitor transmit activity in half-duplex mode). 1 Disable reception of frames while transmitting (normally used for half-duplex mode).
31	LOOP	Internal loopback. When LOOP is set to 1, transmitted frames are looped back internal to the device and the transmit output signals are not asserted. The system clock is substituted for the FEC_TX_CLK when LOOP is set to 1. DRT must be set to zero when setting LOOP to 1.

36.3.4.10 Transmit Control Register (TCR)

TCR bit fields are shown in Figure 468 and described in Table 476. This register is read/write, and is written by the user to configure the transmit block. Bits [2:1] must only be modified when ECR[ETHER_EN] = 0 (that is, during initialization). This register is cleared at system reset.

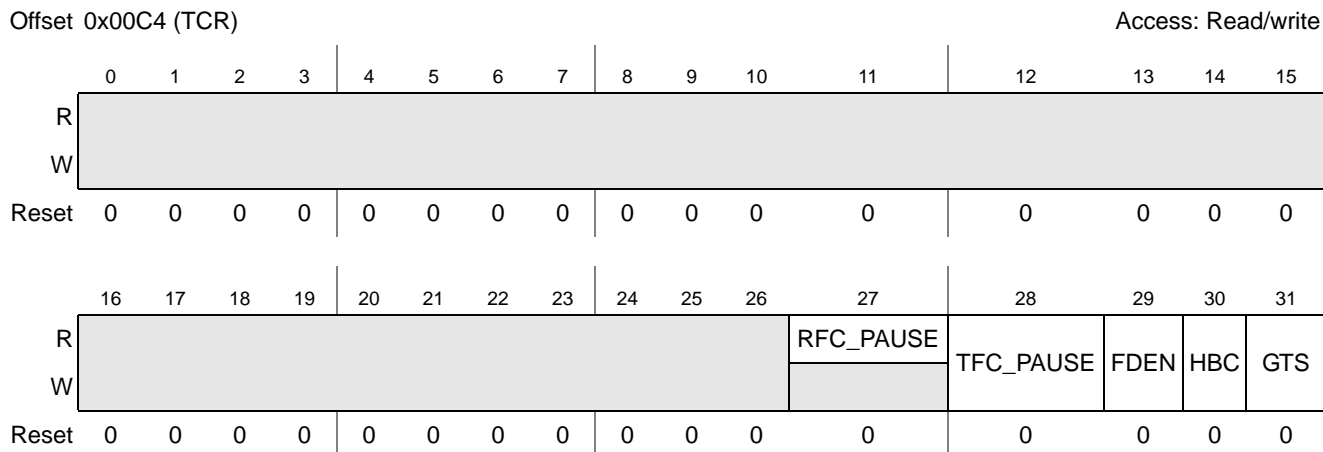


Figure 469. Transmit Control Register (TCR)

Table 477. TCR Field Descriptions

Bits	Name	Description
0–26	—	Reserved read as 0
27	RFC_PAUSE	Receive frame control pause. This read-only status bit is set to 1 when a full-duplex flow control pause frame has been received and the transmitter is paused for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete. 0 Transmitter is not paused 1 Transmitter is paused after reception of full-duplex flow control pause frame
28	TFC_PAUSE	Transmit frame control pause. When this bit is set to 1, a pause frame is transmitted according to the following steps: 1)FEC stops transmission of data frames after the current transmission is complete. 2)The GRA interrupt in the EIR register is asserted. 3)With transmission of data frames stopped, the FEC transmits a MAC control pause frame. 4)The FEC clears the TFC_PAUSE bit and resume transmitting data frames. Note that the FEC can still transmit a MAC control pause frame when the transmitter is paused due to user assertion of GTS or reception of a pause frame. 0 No pause frame is transmitted 1 Pause frame is transmitted
29	FDEN	Full duplex enable. When FDEN is set to 1, frames are transmitted independent of carrier sense and collision inputs. This bit must only be modified when ETHER_EN is cleared. 0 Full duplex is not enabled 1 Full duplex is enabled

Table 477. TCR Field Descriptions (continued)

Bits	Name	Description
30	HBC	Heartbeat control. When HBC is set to 1, the heartbeat check is performed after end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This bit must only be modified when ETHER_EN is cleared. 0 Heartbeat check is not performed after end of transmission 1 Heartbeat check is performed after end of transmission
31	GTS	Graceful transmit stop. When GTS is set to 1, the FEC stops transmission after any frame that is currently being transmitted is complete and the GRA interrupt in the EIR register is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission has completed, a “restart” can be accomplished by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS = 1, transmission stops after the collision. The frame is transmitted again after GTS is cleared. Note: There can be old frames in the transmit FIFO that is transmitted when GTS is reasserted. To avoid this, clear ECR[ETHER_EN] following the GRA interrupt. 0 Graceful transmit stop is not enabled 1 Graceful transmit stop is enabled.

36.3.4.11 Physical Address Low Register (PALR)

The PALR is written by the user, and contains the lower 32 bits (bytes 0,1,2,3) of the 48-bit address used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is also used for bytes 0 through 3 of the 6-byte source address field when transmitting pause frames. This register is unaffected by reset and must be initialized by the user.

PALR bit fields are shown in [Figure 470](#) and described in [Table 478](#).

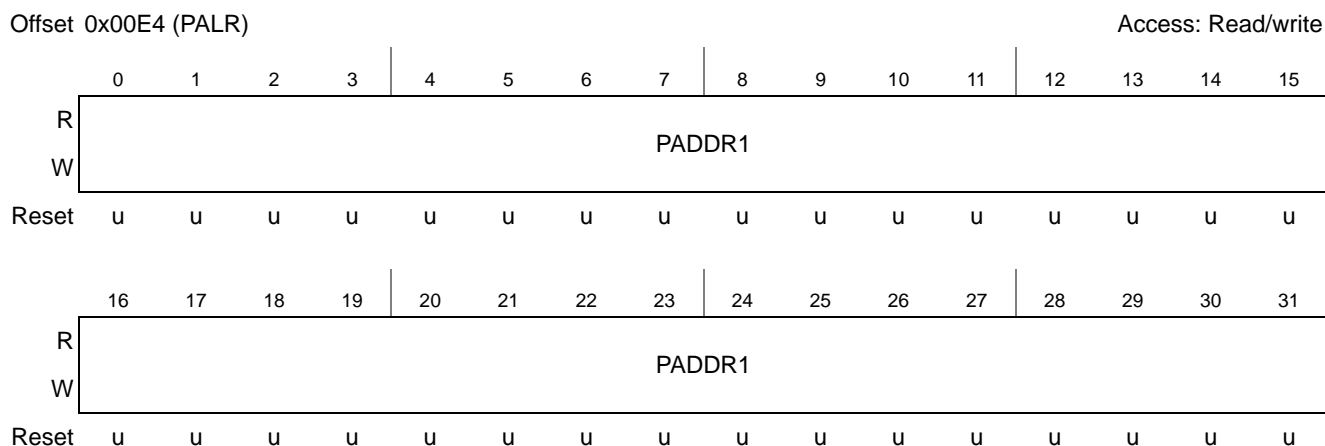


Figure 470. Physical Address Low Register (PALR)

Table 478. PALR Field Descriptions

Bits	Name	Description
0–31	PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8) and 3 (bits 7:0) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.

36.3.4.12 Physical Address Upper Register (PAUR)

The PAUR is written by the user, and contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte source address field when transmitting pause frames. Bits 15:0 of PAUR contain a constant type field (0x8808) used for transmission of pause frames. This register is unaffected by reset, and bits 31:16 must be initialized by the user.

PAUR bit fields are shown in [Figure 471](#) and described in [Table 479](#).

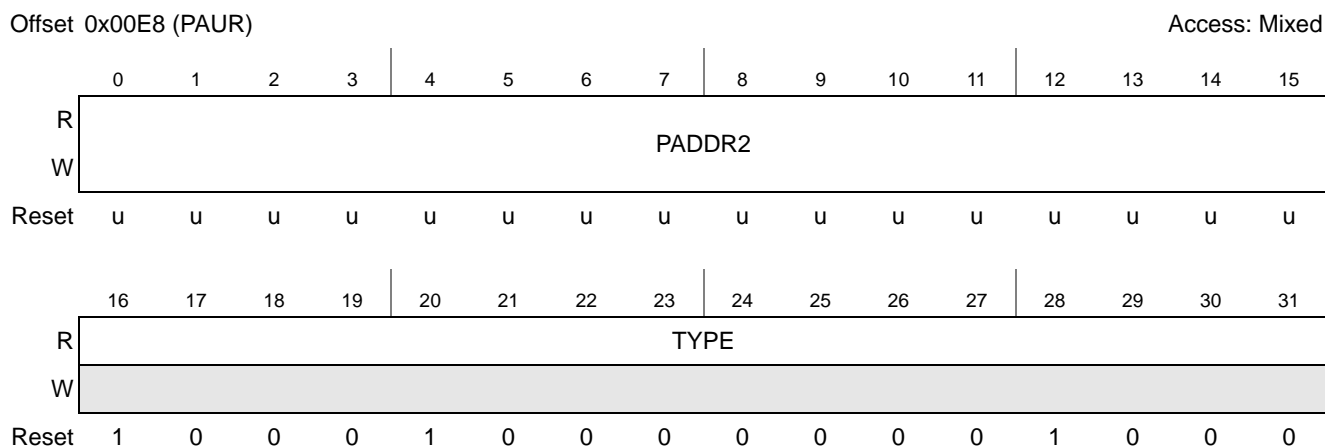


Figure 471. Physical Address Upper Register (PAUR)

Table 479. PAUR Field Descriptions

Bits	Name	Description
0–15	PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.
16–31	TYPE	Type field in pause frames. This field has a constant value of 0x8808.

36.3.4.13 Opcode/Pause Duration Register (OPDR)

OPDR contains the 16-bit Opcode, and 16-bit pause duration fields used in transmission of a pause frame. The Opcode field is a constant value, 0x0001. When another node detects a pause frame, that node pauses transmission for the duration specified in the pause duration field. This register is not reset and must be initialized by the user.

OPDR bit fields are shown in [Figure 472](#) and described in [Table 480](#).

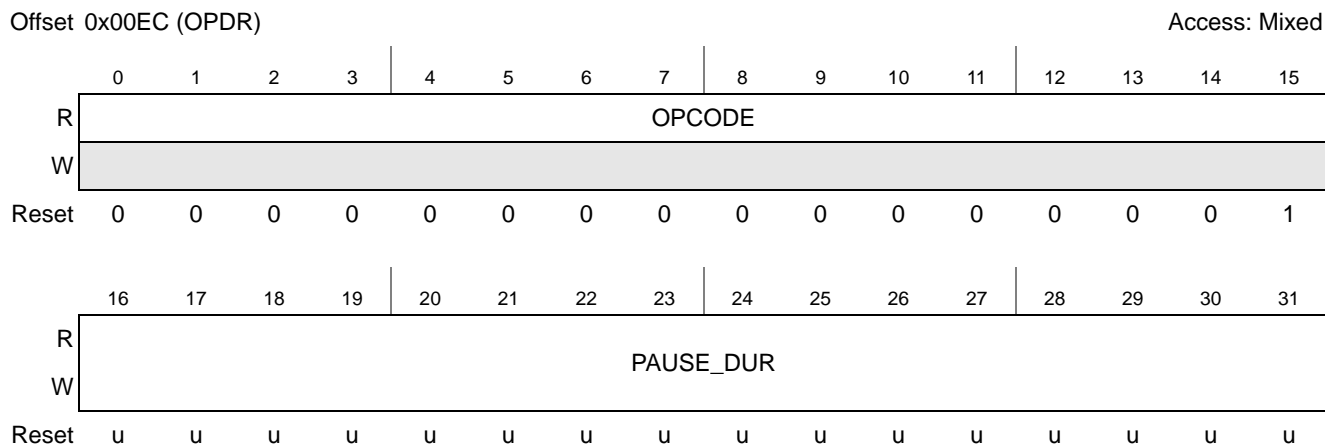


Figure 472. Opcode/Pause Duration Register (OPDR)

Table 480. OPD Field Descriptions

Bits	Name	Description
0–15	OPCODE	Opcode field used in pause frames. These bits are a constant, 0x0001.
16–31	PAUSE_DUR	Pause duration field used in pause frames.

36.3.4.14 Descriptor Individual Address Upper Register (IAUR)

IAUR bit fields are shown in [Figure 473](#) and described in [Table 481](#). The IAUR is written by the user, and contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is unaffected by reset, and must be initialized by the user.

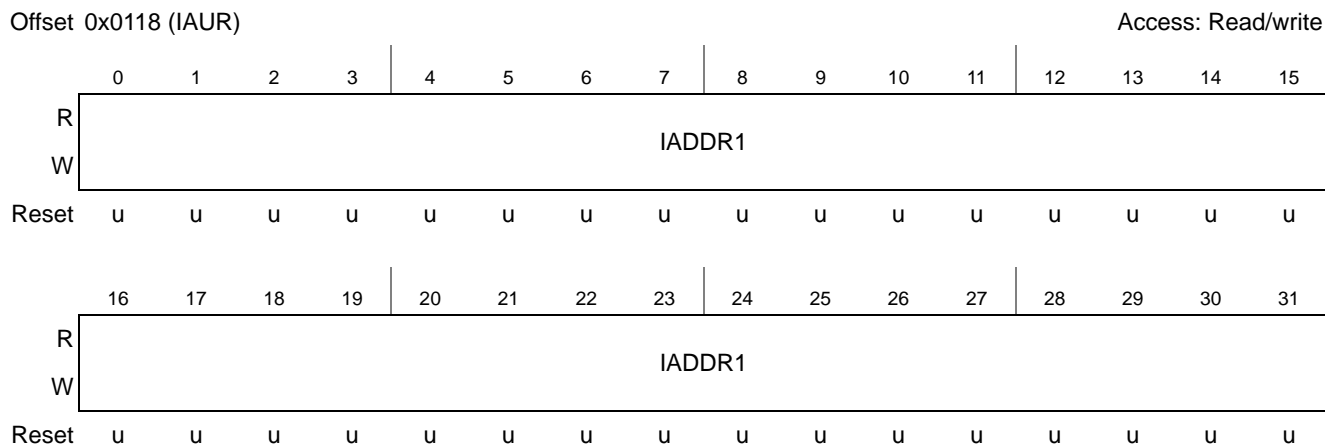


Figure 473. Descriptor Individual Address Register (IAUR)

Table 481. IAUR Field Descriptions

Bits	Name	Description
0–31	IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

36.3.4.15 Descriptor Individual Address Lower Register (IALR)

The IALR is written by the user, and contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is unaffected by reset, and must be initialized by the user.

IAUR bit fields are shown in [Figure 474](#) and described in [Table 482](#).

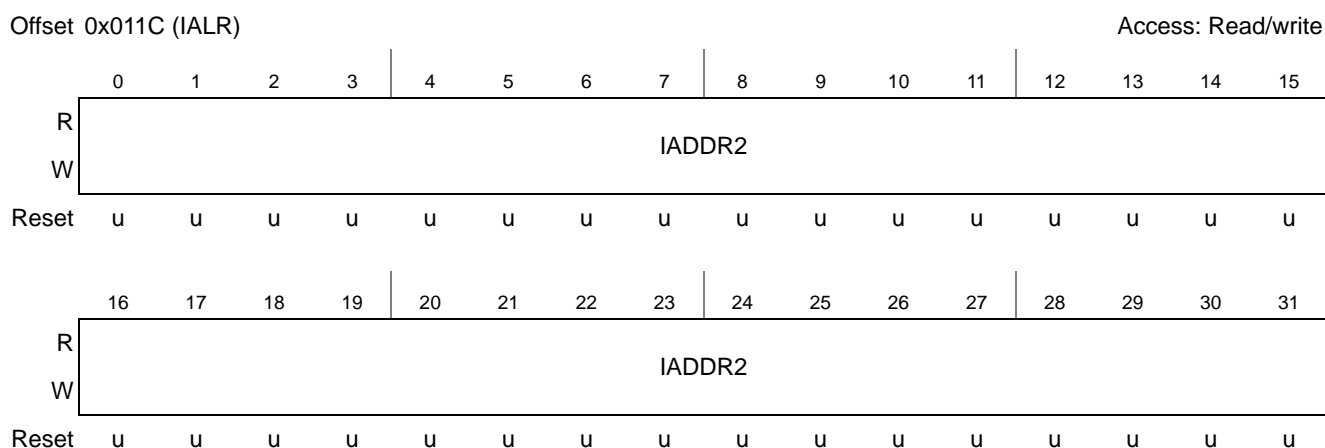


Figure 474. Descriptor Individual Address Lower Register (IALR)

Table 482. IALR Field Descriptions

Bits	Name	Description
0–31	IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

36.3.4.16 Descriptor Group Address Upper Register (GAUR)

The GAUR is written by the user, and contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

GAUR bit fields are shown in [Figure 475](#) and described in [Table 483](#).

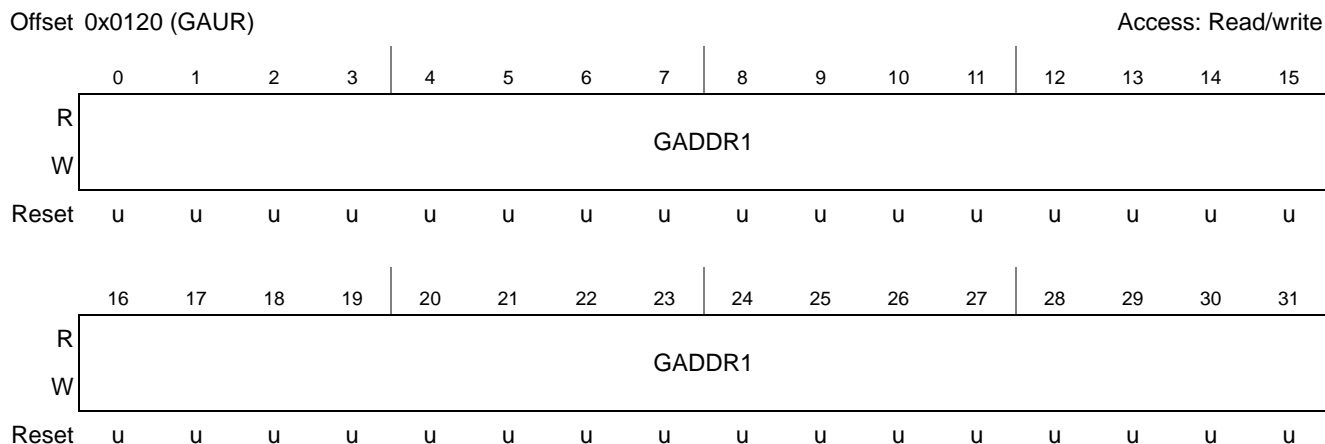


Figure 475. Descriptor Group Upper Address Register (GAUR)

Table 483. GAUR Field Descriptions

Bits	Name	Description
0–31	GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

36.3.4.17 Descriptor Group Address Lower Register (GALR)

The GALR is written by the user, and contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

GALR bit fields are shown in [Figure 476](#) and described in [Table 484](#).

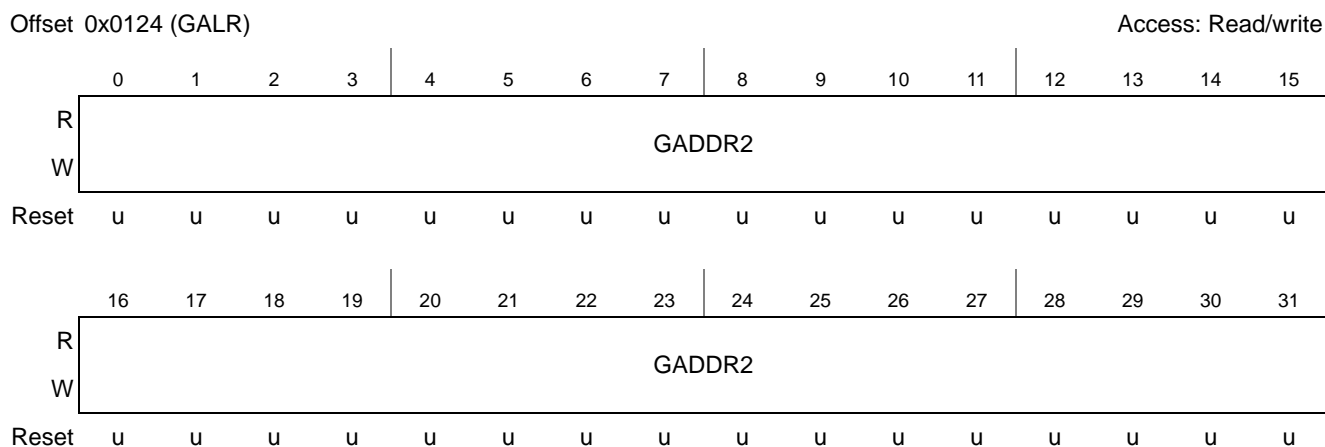


Figure 476. Descriptor Group Lower Address Register (GALR)

Table 484. GALR Field Descriptions

Bits	Name	Description
0–31	GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

36.3.4.18 Transmit FIFO Watermark Register (TFWR)

The TFWR is programmed by the user to control the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows the user to minimize transmit latency (TFWR[1:0] = 0n) or allow for larger bus access latency (TFWR[1:0] = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. In some use cases the byte counts associated with the TFWR field need to be modified to match system requirements, such as the worst-case bus access latency by the transmit data DMA channel.

TFWR bit fields are shown in [Figure 477](#) and described in [Table 485](#).

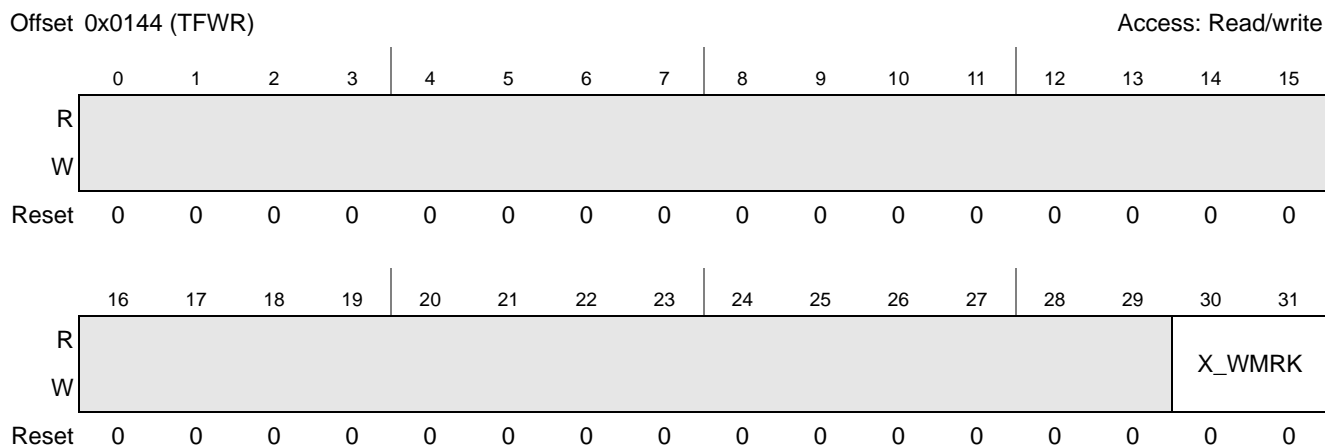


Figure 477. Transmit FIFO Watermark Register (TFWR)

Table 485. TFWR Field Descriptions

Bits	Name	Description
0–29	—	Reserved, read as 0
30–31	X_WMRK	Number of bytes written to transmit FIFO before transmission of a frame begins 0x 64 bytes written 10 128 bytes written 11 192 bytes written

36.3.4.19 FIFO Receive Bound Register (FRBR)

The FRBR register can be read to determine the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR to appropriately divide the available FIFO RAM between the transmit and receive data paths.

FRBR bit fields are shown in [Figure 478](#) and described in [Table 486](#).

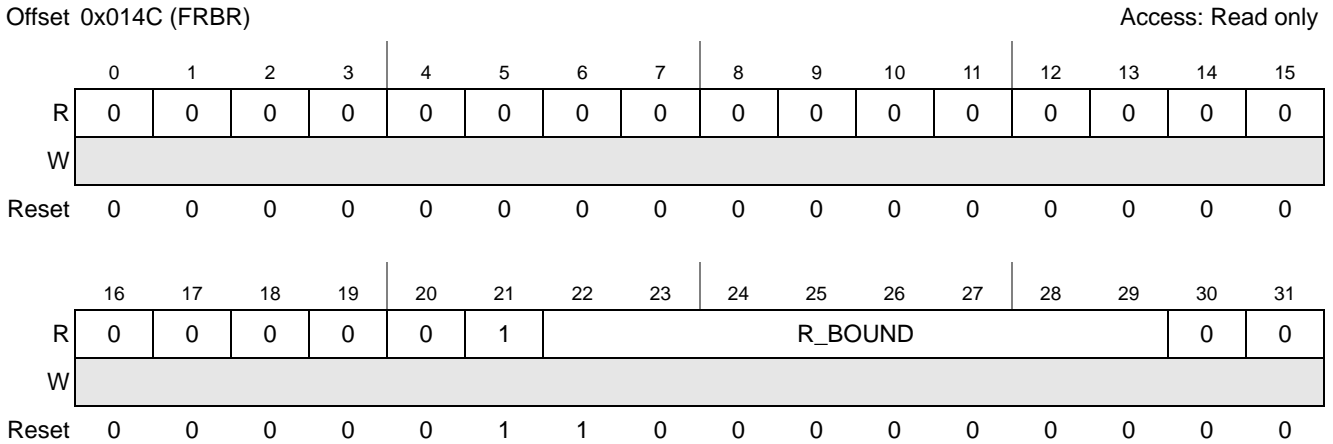


Figure 478. FIFO Receive Bound Register (FRBR)

Table 486. FRBR Field Descriptions

Bits	Name	Description
0–21	—	Reserved, read as 0 (except bit 10, which is read as 1).
22–29	R_BOUND	Read-only. Highest valid FIFO RAM address.
30–31	—	Reserved, read as 0.

36.3.4.20 FIFO Receive Start Register (FRSR)

FRSR is an 8-bit register programmed by the user to indicate the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

The default value of the receive FIFO starting address is 0x40. This is the value assigned by hardware at reset.

FRSR bit assignments are shown in [Figure 479](#) and described in [Table 487](#).

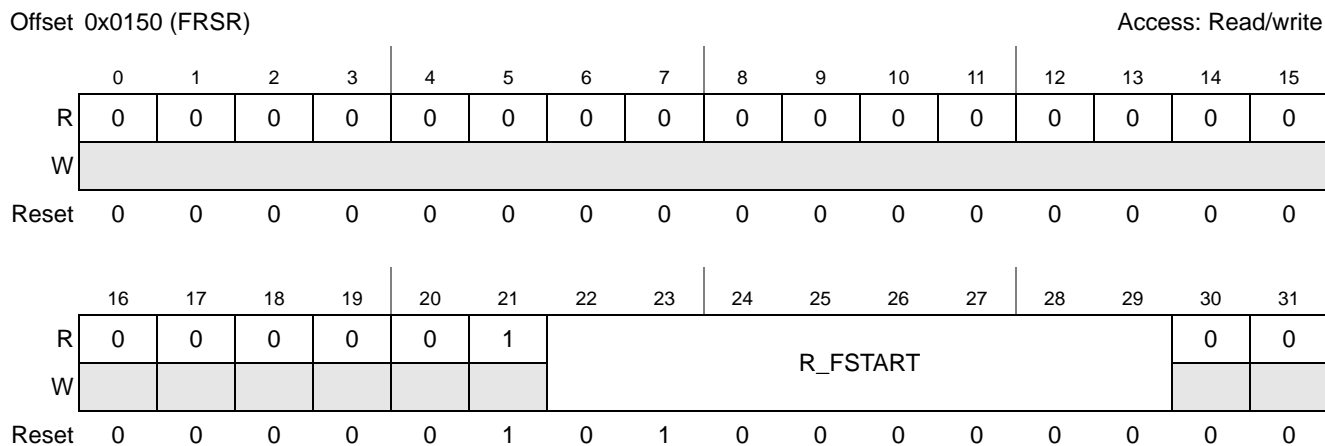


Figure 479. FIFO Receive Start Register (FRSR)

Table 487. FRSR Field Descriptions

Bits	Name	Description
0–21	—	Reserved, read as 0 (except bit 10, which is read as 1).
22–29	R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs.
30–31	—	Reserved, read as 0.

36.3.4.21 Receive Buffer Descriptor Ring Start Register (ERDSR)

The register is written by the user, and provides a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 128-bit aligned (that is, evenly divisible by 16).

ERDSR bit assignments are shown in [Figure 480](#) and described in [Table 488](#).

This register is unaffected by reset and must be initialized by the user prior to operation.

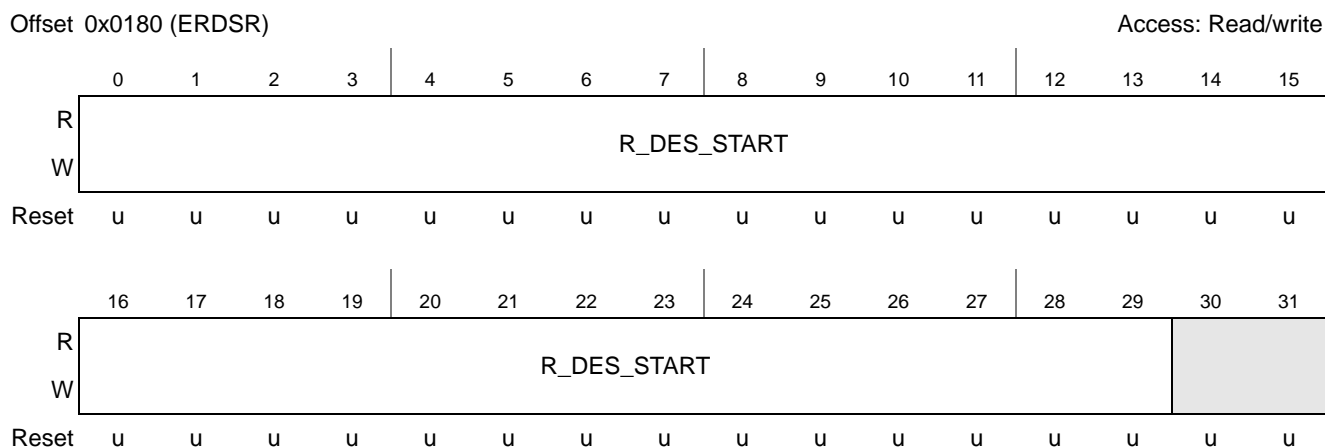


Figure 480. Receive Descriptor Ring Start Register (ERDSR)

Table 488. ERDSR Field Descriptions

Bits	Name	Description
0–29	R_DES_START	Pointer to start of receive buffer descriptor queue.
30–31	—	Reserved, read as 0

36.3.4.22 Transmit Buffer Descriptor Ring Start Register (ETDSR)

ETDSR bit assignments are shown in [Figure 480](#) and described in [Table 488](#). The register is written by the user, and provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 128-bit aligned.

This register is unaffected by reset and must be initialized by the user prior to operation.

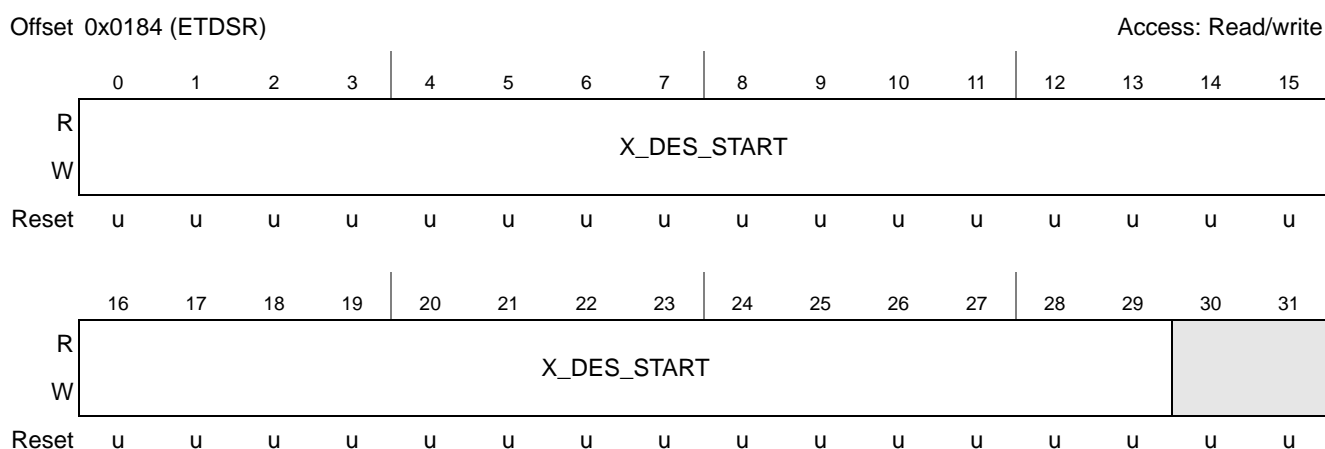


Figure 481. Transmit Buffer Descriptor Ring Start Register (ETDSR)

Table 489. ETDSR Field Descriptions

Bits	Name	Description
0–29	X_DES_START	Pointer to start of transmit buffer descriptor queue.
30–31	—	Reserved, read as 0

36.3.4.23 Maximum Receive Buffer Size Register (EMRBR)

The EMRBR, shown in [Figure 482](#) and [Table 490](#), is a user-programmable register which dictates the maximum size of all receive buffers. Note that because receive frames is truncated at 2k-1(2047) bytes, bits 31-11 are not used. The programmed value accounts for the fact that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. The EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low, and hence only bits 10-4 are actually used. To minimize bus utilization (descriptor fetches) it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR is unaffected by reset, and must be initialized by the user.

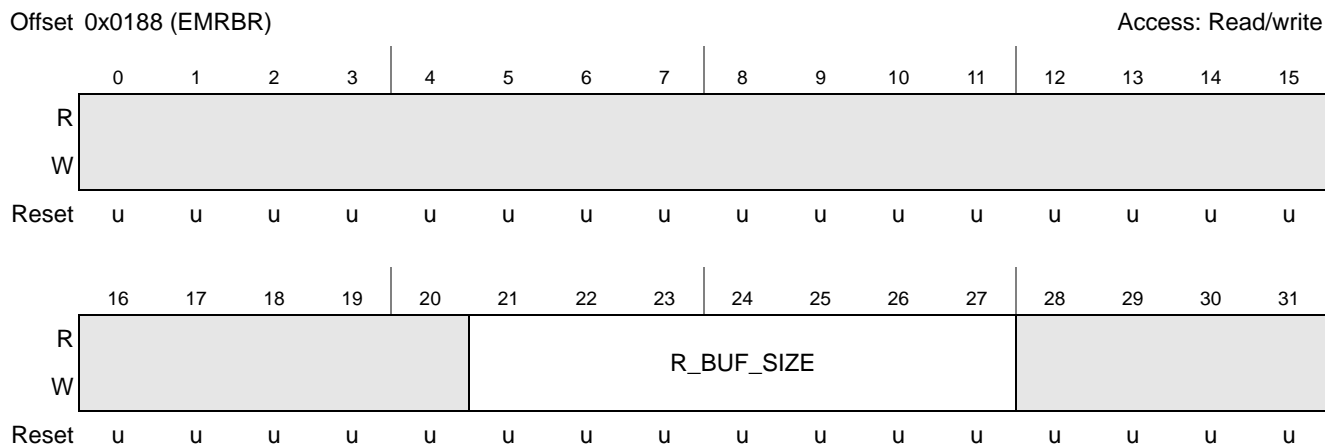


Figure 482. Receive Buffer Size Register (EMRBR)

Table 490. EMRBR Field Descriptions

Bits	Name	Description
0–20	—	Reserved, is written to 0 by the host processor.
21–27	R_BUF_SIZE	Receive buffer size.
28–31	—	Reserved, is written to 0 by the host processor.

36.4 Functional Description

This section provides a detailed description of the functions of the FEC including:

- Network interface options
- Frame transmission and reception
- Full-duplex flow control
- Internal and external loopback

36.4.1 Network Interface Options

The FEC supports an MII interface for 10/100 Mbps Ethernet. The interface mode is selected by the RCR[MII_MODE] bit.

In MII mode (RCR[MII_MODE] = 1), there are 18 signals defined by the IEEE 802.3 standard and supported by the FEC. These signals are shown in [Table 491](#) below.

Table 491. MII Mode Signal Configuration

Signal Description	FEC Signal Name	Direction
Transmit clock	FEC_TX_CLK	In
Transmit enable	FEC_TX_EN	Out

Table 491. MII Mode Signal Configuration (continued)

Signal Description	FEC Signal Name	Direction
Transmit data	FEC_TXD[3:0]	Out
Transmit error	FEC_TX_ER	Out
Collision	FEC_COL	In
Carrier sense	FEC_CR_S	In
Receive clock	FEC_RX_CLK	In
Receive data valid	FEC_RX_DV	In
Receive data	FEC_RXD[3:0]	In
Receive error	FEC_RX_ER	In
Management data clock	FEC_MDC	Out
Management data input/output	FEC_MDIO	I/O

36.4.2 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set to 1 and data appears in the transmit FIFO, the FEC is able to transmit onto the network.

When the transmit FIFO fills to the watermark (defined by the TFWR register), the FEC transmit logic asserts FEC_TX_EN and start transmitting the preamble (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller postpones transmission if the network is busy (that is, the carrier sense signal FEC_CR_S is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense remains inactive for 60 bit periods. If so, the transmission begins after waiting an additional 36 bit periods (96 bit periods after carrier sense originally became inactive). See [Section 36.4.2.3, “Transmission Error Handling”](#) for more details.

If a collision occurs during transmission of a frame in half-duplex mode, the Ethernet controller follows the specified backoff procedures (see [Section 36.4.2.2, “Collision Handling”](#)) and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, if the TC bit is set in the transmit frame control word then a 32-bit cyclic redundancy check (CRC) known as the frame check sequence (FCS) is appended. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set to 1).

Both buffer (TXB) and frame (TXF) interrupts can be generated as determined by the settings in the EIMR.

The transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, and XFIFO_UN. If the transmit frame length exceeds MAX_FL bytes the BABT interrupt is asserted: however, the entire frame is transmitted (no truncation).

Transmission is paused by setting the graceful transmit stop (GTS) bit in the TCR register. When the TCR[GTS] is set to 1, the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame either finishes or terminates with a collision. After the transmitter has stopped the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

The Ethernet controller transmits bytes LSB first.

36.4.2.1 Transmit Inter-Packet Gap (IPG) Time

The minimum inter-packet gap (IPG) time for back-to-back transmission is 96 bit periods. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96-bit-period IPG counter. Frame transmission begins 96 bit periods after carrier sense is negated if it stays negated for at least 60 bit periods. If carrier sense is asserted during the last 36 bit periods, it is ignored and a collision occurs.

36.4.2.2 Collision Handling

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit periods, transmitting a jam pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit periods, the retry process is initiated. The transmitter waits a random number of slot periods, where one slot period is 512 bit periods. If a collision occurs after 512 bit periods, then no retransmission is performed and the end of frame buffer is closed with a late collision (LC) error indication.

36.4.2.3 Transmission Error Handling

The Ethernet controller reports frame transmission error conditions using the FEC RxBDs, the EIR register, and the MIB block counters

There are four types of transmission errors:

- Transmitter underrun
- Retransmission attempts limit expired
- Late collision
- Heartbeat

The FEC's procedures for handling these errors are described in the following subsections.

36.4.2.3.1 Transmitter Underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error, then stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the EIR and the UN interrupt is

asserted (if enabled in the EIMR register). The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

36.4.2.3.2 Retransmission Attempts Limit Expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the RL bit is set in the EIR. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

The “RL” interrupt is asserted if enabled in the EIMR register.

36.4.2.3.3 Late Collision

When a collision occurs after the slot time (512 bits starting at the preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the LC bit is set in the EIR register. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

The LC interrupt is asserted if enabled in the EIMR register.

36.4.2.3.4 Heartbeat

Some transceivers have a self-test feature called “heartbeat” or “signal quality error.” To signify a good self-test, the transceiver indicates a collision to the FEC within 4 microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.

If the HBC bit is set in the TCR register and the heartbeat condition is not detected by the FEC after a frame transmission, then a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets the HB bit in the EIR register, and generates the HBERR interrupt if it is enabled.

36.4.3 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking.

When the driver enables the FEC receiver by setting ECR[ETHER_EN] to 1, it immediately starts processing receive frames. When FEC_RX_DV asserts, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the frame is processed by the receiver. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit periods of RX_D0 following assertion of FEC_RX_DV are ignored. Following the first 16 bit periods the data sequence is checked for alternating 1/0s. If a 0b11 or 0b00 data sequence is detected during bit periods 17 to 21, the remainder of the frame is ignored. After bit period 21, the data sequence is monitored for a valid SFD (11). If a 0b00 is detected, the frame is rejected. When a 0b11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes can occur, but if a 0b00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame have been received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data has been received and if address recognition has not rejected the frame, the receive FIFO is signaled that the frame is accepted and can be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Thus no collision fragments are presented to the user except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV and TR status bits, and the frame length. See [Section 36.4.3.3, “Reception Error Handling”](#) for more details.

Receive Buffer (RXB) and Frame Interrupts (RXF) can be generated if enabled by the EIMR register. The only receive error interrupt is babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See [Section 36.5.2.3, “Ethernet Receive Buffer Descriptor \(RxBD\)”](#) for more details.

When the receive frame is complete, the FEC sets the L bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E bit. The Ethernet controller next generates a maskable interrupt (RXF bit in EIR, maskable by RXF bit in EIMR), indicating that a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

36.4.3.1 Receive Inter-Packet Gap (IPG) Time

The receiver receives back-to-back frames with a minimum spacing of at least 28 bit periods. If an inter-packet gap between receive frames is less than 28 bit periods, the second frame may be discarded by the receiver.

36.4.3.2 Ethernet Address Recognition

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in [Figure 483](#) illustrates the address recognition decisions made by the receive block, while [Figure 484](#) illustrates the decisions made by the microcontroller.

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the DA field.

If the DA is a broadcast address, and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in [Figure 483](#). Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in [Figure 484](#).

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame. The hash algorithm is described in [Section 36.4.3.2.1, “Hash Algorithm”](#).

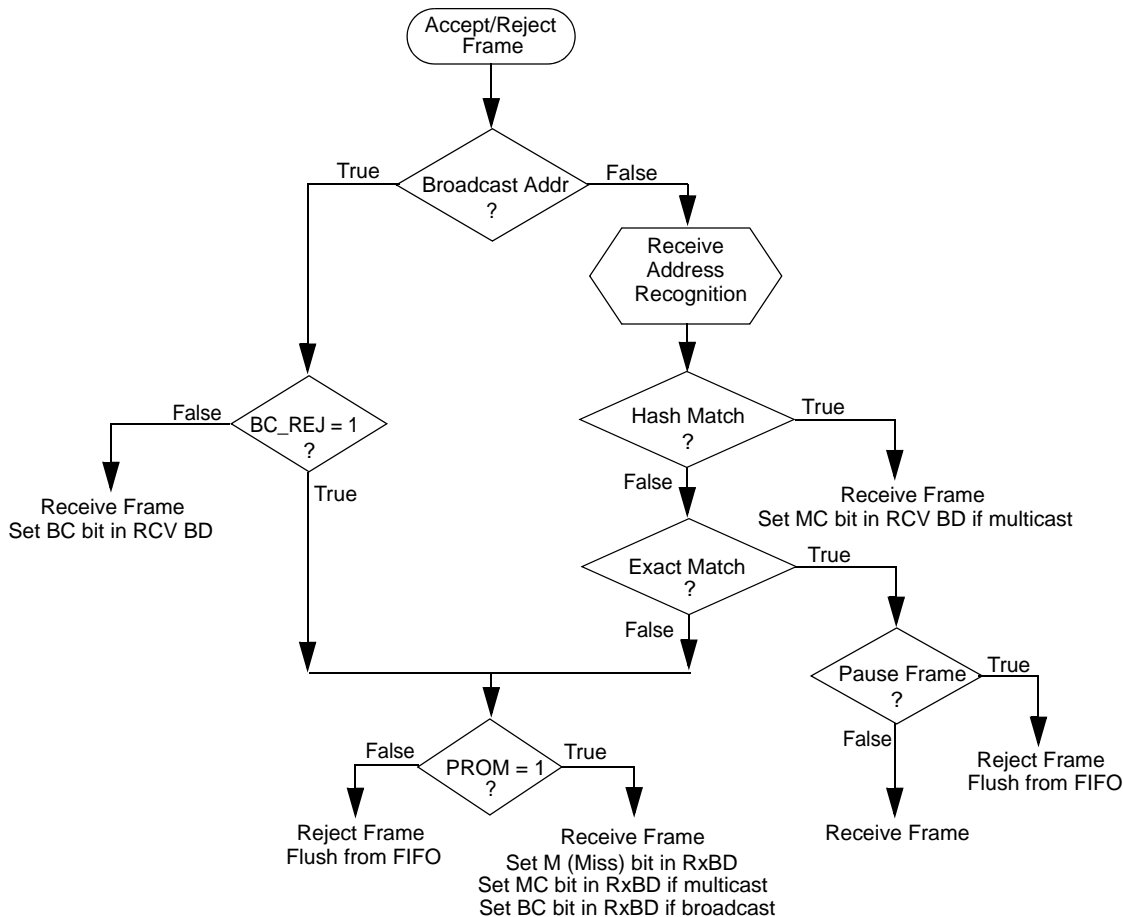
If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines that the received frame is a valid pause frame, then the frame is rejected. Note the receiver detects a pause frame with the DA field set to either the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that the user programs in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR (the hash algorithm is described in [Section 36.4.3.2.1, “Hash Algorithm”](#)). In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on pause frame detection, shown in [Figure 483](#).

If neither a hash match (group or individual), nor an exact match occur, then if promiscuous mode is enabled ($\text{RCR}[\text{PROM}] = 1$), then the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

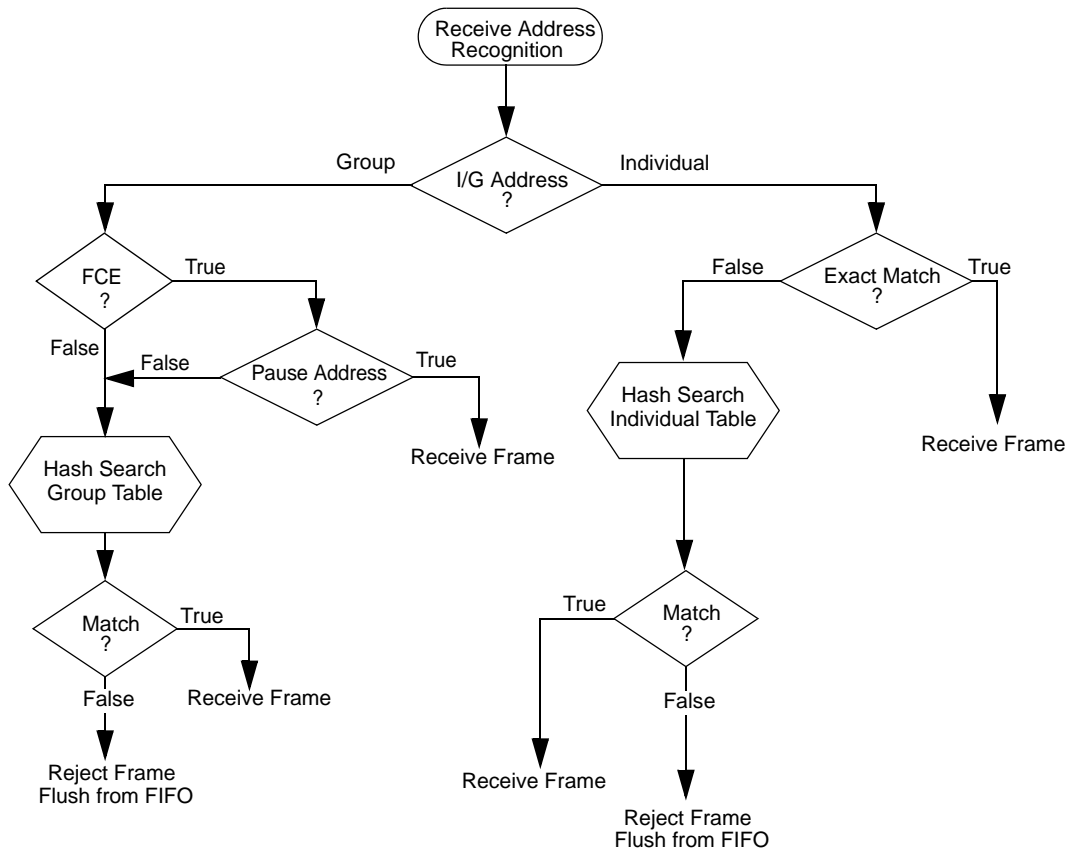
Similarly, if the DA is a broadcast address, broadcast reject ($\text{RCR}[\text{BC_REJ}]$) is asserted, and promiscuous mode is enabled, then the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



NOTES:
 BC_REJ—Field in RCR register (BroadCast REJect)
 PROM—Field in RCR register (PROMiscuous mode)
 Pause Frame—Valid PAUSE frame received

Figure 483. Ethernet Address Recognition—Receive Block Decisions



NOTES:
 FCE - field in RCR register (Flow Control Enable)
 I/G - Individual/Group bit in Destination Address (least significant bit in first byte received in MAC frame)

Figure 484. Ethernet Address Recognition—Microcode Decisions

36.4.3.2.1 Hash Algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, which are represented by 64 bits stored in GAUR, GALR (group address hash match) or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects GAUR (MSB = 1) or GALR (MSB = 0). The least significant five bits of the hash result select the bit within the selected register. If the CRC generator selects a bit that is set in the hash table, the frame is accepted; otherwise, it is rejected. For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized by the user. The CRC32 polynomial to use in computing the hash is shown in Equation :

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Table 492 shows example destination addresses and corresponding hash values is included below for reference.

Table 492. Destination Address to 6-Bit Hash

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
65:FF:FF:FF:FF:FF	0x0	0
55:FF:FF:FF:FF:FF	0x1	1
15:FF:FF:FF:FF:FF	0x2	2
35:FF:FF:FF:FF:FF	0x3	3
B5:FF:FF:FF:FF:FF	0x4	4
95:FF:FF:FF:FF:FF	0x5	5
D5:FF:FF:FF:FF:FF	0x6	6
F5:FF:FF:FF:FF:FF	0x7	7
DB:FF:FF:FF:FF:FF	0x8	8
FB:FF:FF:FF:FF:FF	0x9	9
BB:FF:FF:FF:FF:FF	0xA	10
8B:FF:FF:FF:FF:FF	0xB	11
0B:FF:FF:FF:FF:FF	0xC	12
3B:FF:FF:FF:FF:FF	0xD	13
7B:FF:FF:FF:FF:FF	0xE	14
5B:FF:FF:FF:FF:FF	0xF	15
27:FF:FF:FF:FF:FF	0x10	16
07:FF:FF:FF:FF:FF	0x11	17
57:FF:FF:FF:FF:FF	0x12	18
77:FF:FF:FF:FF:FF	0x13	19
F7:FF:FF:FF:FF:FF	0x14	20
C7:FF:FF:FF:FF:FF	0x15	21
97:FF:FF:FF:FF:FF	0x16	22
A7:FF:FF:FF:FF:FF	0x17	23
99:FF:FF:FF:FF:FF	0x18	24
B9:FF:FF:FF:FF:FF	0x19	25

Table 492. Destination Address to 6-Bit Hash (continued)

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
F9:FF:FF:FF:FF:FF	0x1A	26
C9:FF:FF:FF:FF:FF	0x1B	27
59:FF:FF:FF:FF:FF	0x1C	28
79:FF:FF:FF:FF:FF	0x1D	29
29:FF:FF:FF:FF:FF	0x1E	30
19:FF:FF:FF:FF:FF	0x1F	31
D1:FF:FF:FF:FF:FF	0x20	32
F1:FF:FF:FF:FF:FF	0x21	33
B1:FF:FF:FF:FF:FF	0x22	34
91:FF:FF:FF:FF:FF	0x23	35
11:FF:FF:FF:FF:FF	0x24	36
31:FF:FF:FF:FF:FF	0x25	37
71:FF:FF:FF:FF:FF	0x26	38
51:FF:FF:FF:FF:FF	0x27	39
7F:FF:FF:FF:FF:FF	0x28	40
4F:FF:FF:FF:FF:FF	0x29	41
1F:FF:FF:FF:FF:FF	0x2A	42
3F:FF:FF:FF:FF:FF	0x2B	43
BF:FF:FF:FF:FF:FF	0x2C	44
9F:FF:FF:FF:FF:FF	0x2D	45
DF:FF:FF:FF:FF:FF	0x2E	46
EF:FF:FF:FF:FF:FF	0x2F	47
93:FF:FF:FF:FF:FF	0x30	48
B3:FF:FF:FF:FF:FF	0x31	49
F3:FF:FF:FF:FF:FF	0x32	50
D3:FF:FF:FF:FF:FF	0x33	51
53:FF:FF:FF:FF:FF	0x34	52
73:FF:FF:FF:FF:FF	0x35	53
23:FF:FF:FF:FF:FF	0x36	54
13:FF:FF:FF:FF:FF	0x37	55
3D:FF:FF:FF:FF:FF	0x38	56
0D:FF:FF:FF:FF:FF	0x39	57
5D:FF:FF:FF:FF:FF	0x3A	58

Table 492. Destination Address to 6-Bit Hash (continued)

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
7D:FF:FF:FF:FF:FF	0x3B	59
FD:FF:FF:FF:FF:FF	0x3C	60
DD:FF:FF:FF:FF:FF	0x3D	61
9D:FF:FF:FF:FF:FF	0x3E	62
BD:FF:FF:FF:FF:FF	0x3F	63

36.4.3.3 Reception Error Handling

The Ethernet controller reports frame reception error conditions using the FEC RxBDs, the EIR register, and the MIB block counters

There are five types of reception errors:

- Overrun
- Non-octet (dribbling bits)
- CRC
- Frame-length violation
- Truncation

These are described in the following subsections.

36.4.3.3.1 Overrun

If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the RxBD. All subsequent data in the frame is discarded, and subsequent frames can also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. This frame must be discarded by the driver.

36.4.3.3.2 Non-Octet (Dribbling Bits)

The Ethernet controller handles up to seven dribbling bits when the receive frame terminates past a non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, then the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, then no error is reported.

36.4.3.3.3 CRC

When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the RxBD. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

36.4.3.3.4 Frame Length Violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and the LG bit in the end of frame RxBD is set. The frame is not truncated unless the frame length exceeds 2047 bytes).

36.4.3.3.5 Truncation

When the receive frame length exceeds 2047 bytes, the frame is truncated and the TR bit is set in the RxBD.

36.4.4 Full-Duplex Flow Control

Full-duplex flow control allows the user to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, FEC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] asserted) and flow control enable (RCR[FCE]) must be asserted. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in Table 493. In addition, the receive status associated with the frame indicates that the frame is valid.

Table 493. Pause Frame Field Specification

48-bit destination address	0x0180_C200_0001 or physical address
48-bit source address	Any
16-bit type	0x8808
16-bit opcode	0x0001
16-bit pause duration	0x0000–0xFFFF

Pause frame detection is performed by the receiver and microcontroller modules. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is asserted by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. Note that the pause timer makes use of the transmit backoff timer hardware, which is used for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments after every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing FEC data frame transmission to resume. Note that the receive flow control pause (TCR[RFC_PAUSE]) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and the user must assert flow control pause (TCR[TFC_PAUSE]). On assertion of transmit flow control pause (TCR[TFC_PAUSE]), the transmitter asserts TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts. Following EIR[GRA] assertion, the pause frame is transmitted. On completion of pause frame transmission, flow control pause (TCR[TFC_PAUSE]) and TCR[GTS] are cleared internally.

The user must specify the desired pause duration in the OPD register.

Note that when the transmitter is paused due to receiver/microcontroller pause frame detection, transmit flow control pause (TCR[TFC_PAUSE]) still can be asserted and causes the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

36.4.5 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of the LOOP and DRT bits in the RCR register and the FDEN bit in the TCR register.

For both internal and external loopback set FDEN = 1.

For internal loopback set RCR[LOOP] = 1 and RCR[DRT] = 0. FEC_TX_EN and FEC_TX_ER does not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit and receive data being transferred to and from external memory via DMA. It can be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For external loopback set RCR[LOOP] = 0, RCR[DRT] = 0 and configure the external transceiver for loopback.

36.5 Initialization/Application Information

36.5.1 Initialization Sequence

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC, and what locations the user must initialize prior to enabling the FEC.

36.5.1.1 Hardware Controlled Initialization

In the FEC, registers and control logic that generate interrupts are reset by hardware. A hardware reset negates output signals and resets general configuration bits.

Other registers are reset when the ECR[ETHER_EN] bit is cleared, as indicated in [Table 494](#). ECR[ETHER_EN] is cleared by a hard reset or can be cleared by software to halt operation. By clearing ECR[ETHER_EN], the configuration control registers such as the TCR and RCR do not reset, but the entire data path resets.

Table 494. Effect on FEC of Clearing ECR[ETHER_EN]

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated

Table 494. Effect on FEC of Clearing ECR[ETHER_EN] (continued)

Register/Machine	Reset Value
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

36.5.1.2 User Initialization (Prior to Asserting ECR[ETHER_EN])

The user must initialize portions of the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The order of initializations is not important except explicit mentioned.

The FEC and FEC FIFO/DMA registers requiring initialization are defined in [Table 495](#).

Table 495. Registers Requiring Initialization before Setting ECR[ETHER_EN]

Register	Location (FEC or FIFO/DMA)	Comments
EIMR	FEC	Requires initialization
EIR	FEC	Must be cleared by writing 0xFFFF_FFFF
TFWR	FEC	Optional
IALR / IAUR	FEC	—
GAUR / GALR	FEC	—
PALR / PAUR	FEC	—
OPD	FEC	Only needed for full-duplex flow control
RCR	FEC	—
TCR	FEC	—
MSCR	FEC	Optional
MIB counters	FEC	Must be cleared
FRSR	FIFO/DMA	Initialization is optional
EMRBR	FIFO/DMA	—
ERDSR	FIFO/DMA	—
ETDSR	FIFO/DMA	—
Transmit Descriptor ring	FIFO/DMA	Must be emptied
Receive Descriptor ring	FIFO/DMA	Must be emptied

36.5.1.3 Microcontroller Initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, the hardware is ready for operation.

The microcontroller initialization sequence follows:

1. Initialize BackOff Random Number Seed
2. Activate Receiver
3. Activate Transmitter

4. Clear Transmit FIFO
5. Clear Receive FIFO
6. Initialize Transmit Ring Pointer
7. Initialize Receive Ring Pointer
8. Initialize FIFO Count Register

36.5.1.4 User Initialization (after asserting ECR[ETHER_EN])

After asserting ECR[ETHER_EN], the user can set up the buffer/frame descriptors and write to the TDAR and RDAR. See [Section 36.5.2, “Buffer Descriptors”](#) for more details.

36.5.2 Buffer Descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors (BD). It is followed by a detailed description of the receive and transmit descriptor fields.

36.5.2.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. Associated with each buffer is a buffer descriptor (BD) which contains a starting address (pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read in by the FEC DMA engine.

Software “produces” buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit “produces” the buffer. Software writing to either the TDAR or RDAR tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and “consumes” the buffers after they have been produced. After the data DMA is complete and the buffer descriptor status bits have been written by the DMA engine, the RxBD[E] or TxBD[R] bit is cleared by hardware to signal the buffer has been “consumed.” Software can poll the BDs to detect when the buffers have been consumed or can rely on the buffer/frame interrupts. These buffers can then be processed by the driver and returned to the free list.

The ECR[ETHER_EN] signal operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before the ECR[ETHER_EN] bit is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The last buffer descriptor in each ring is defined by the Wrap (W) bit. When set, W indicates that the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively.

NOTE

Buffer descriptor rings must start on a 128-bit boundary.

36.5.2.2 Ethernet Transmit Buffer Descriptor (TxBD)

Figure 485 shows the transmit buffer descriptor format. Table 496 describes the TxBD fields.

Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block. See Section 36.3.3, “Message Information Block (MIB) Counters Memory Map”, for more details.

Offset	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
0x0000	R	TO1	W	TO2	L	TC	ABC	PTP								
	Data Length															
0x0004	Tx Data Buffer Pointer A[31:16]															
	Tx Data Buffer Pointer A[15:0]															

Figure 485. Transmit Buffer Descriptor (TxBD)

Table 496. Transmit Buffer Descriptor Field Definitions

Offset	Field	Description
0x0000	31 R	Ready. Written by the FEC and the user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD can be written by the user after this bit is set.
	0x0000	30 TO1
0x0000	29 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
0x0000	28 TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	27 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
0x0000	26 TC	Tx CRC. Written by user (only valid if L = 1). 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
0x0000	25 ABC	Append bad CRC. Written by user (only valid if L = 1). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value).
0x0000	24 PTP	PTP. Written by user (only valid if L = 1). 0 No effect 1 The frame is an IEEE1588 PTP frame. FEC will inform IEEE1588 Assist module to capture the timestamp for this frame in case of successful transmission. If unmasked, an interrupt will be issued by the IEEE1588 Assist module after the timestamp is available.

Table 496. Transmit Buffer Descriptor Field Definitions (continued)

Offset	Field	Description
0x0000	23–16	Reserved.
0x0000	15–0 Data Length	Data Length, written by user. Data length is the number of octets the FEC transmits from this BD's data buffer. It is never modified by the FEC. Bits [10:0] are used by the DMA engine, bits[15:11] are ignored.
0x0004	31–0 A	Tx data buffer pointer ¹

¹ The transmit buffer pointer contains the address of the associated data buffer, which is 16 byte aligned. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

36.5.2.2.1 Driver/DMA Operation with Transmit Buffer Descriptors

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. After the software driver has set up the buffers for a frame, it sets up the corresponding BDs. In the TxBD the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword, and the buffer pointer in the second longword. The last step in setting up the BDs for a transmit frame is to set the R bit in the first BD for the frame. The driver follows that with a write to TDAR which triggers the FEC to poll the next BD in the ring.

The Ethernet controller confirms transmission by clearing the ready bit (R bit) when DMA of the buffer is complete.

36.5.2.2.1.1 Transmit Frame in Multiple Buffers

Typically a transmit frame is divided between multiple buffers. For example, it is possible to have an application payload in one buffer, TCP header in a 2nd buffer, IP header in a 3rd buffer, and Ethernet/IEEE 802.3 header in a 4th buffer. The FEC does not prepend the Ethernet header (Destination Address, Source Address, Length/Type field(s)), so this must be provided by the driver in one of the transmit buffers. The FEC can append the Ethernet CRC to the frame. Whether the CRC is appended by the FEC or by the driver is determined by the TC bit in the transmit BD, which must be set by the driver.

The driver (TxBD software producer) sets up Tx BDs in such a way that a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, first the BD's are initialized with pointer, length and control (W, L, TC, ABC) and then the TxBD[R] bits are set to 1 in *reverse order* (3rd, 2nd, 1st BD) to insure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the 2nd was made available, potentially causing a transmit FIFO underrun.

In the FEC, the DMA is notified by the driver that new transmit frame(s) are available by writing to the TDAR register. When this register is written to (data value is not significant) the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers, until a transmit BD is encountered with the R bit = 0. At this point the FEC polls this BD one more time. If the R bit = 0 the second time, then the RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

36.5.2.3 Ethernet Receive Buffer Descriptor (RxBD)

Figure 486 shows the receive buffer descriptor format. Table 497 describes the RxBD fields.

Offset	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
0x0000	E	RO1	W	RO2	L	PTP		M	B	MC	LG	NO		CR	OV	TR
	Data Length															
0x0004	Rx Data Buffer Pointer A[31:16]															
	Rx Data Buffer Pointer A[15:0]															

Figure 486. Receive Buffer Descriptor (RxBD)

Table 497. Receive Buffer Descriptor Field Definitions

Offset	Field	Description
0x0000	31 E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
0x0000	30 RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	29 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.
0x0000	28 RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	27 L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
0x0000	26 PTP	PTP frame received. Written by the FEC. 0 Received frame is not an IEEE1588 PTP frame. 1 Received frame is an IEEE1588 PTP frame. FEC has informed IEEE1588 Assist module to capture the timestamp for this frame. If unmasked, an interrupt will be issued by the IEEE1588 Assist module after the timestamp is available. This bit is valid only if the L-bit is set.
0x0000	25	Reserved.

Table 497. Receive Buffer Descriptor Field Definitions (continued)

Offset	Field	Description
0x0000	24 M	Miss. Written by the FEC. This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a “miss” by the internal address recognition. Thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
0x0000	23 BC	is set if the DA is broadcast (FF:FF:FF:FF:FF:FF).
0x0000	22 MC	is set if the DA is multicast and not BC.
0x0000	21 LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
0x0000	20 NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set the CR bit is not set.
0x0000	19	Reserved
0x0000	18 CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
0x0000	17 OV	Overflow. Written by the FEC. A receive FIFO overflow occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and is zero. This bit is valid only if the L-bit is set.
0x0000	16 TR	is set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set the frame is discarded and the other error bits ignored as they can be incorrect.
0x0000	15–0 Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD’s data buffer if L = 0 (the value is equal to EMRBR), or the length of the frame including CRC if L = 1. It is written by the FEC once as the BD is closed.
0x0004	31–0	RX data buffer pointer ¹

¹ The receive buffer pointer, which contains the address of the associated data buffer, must always be divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

36.5.2.3.1 Driver/DMA Operation with Receive Buffer Descriptors

Unlike the transmit case, the length of the receive frame is unknown by the driver ahead of time. Therefore the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxB software producer) sets up some number of “empty” buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L (1 indicates last buffer in frame) bit, the frame status bits (if L = 1) and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end of frame buffers, the L=1 bit is set in the receive BD, and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame is discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not just the length of the last buffer.

For simplicity the driver can assign the default receive buffer length to be large enough to contain an entire frame, keeping in mind that a malfunction on the network or out-of-specification implementation could result in giant frames. Frames of 2 KB (2048 bytes) or larger are truncated by the FEC at 2047 bytes, so software never sees a receive frame larger than 2047 bytes.

As in the transmit case, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received the FEC fills receive buffers and update the associated BDs, then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit is cleared, it polls this BD once more. If E is still cleared, then the FEC stops reading receive BDs until the driver writes to RDAR.

NOTE

Whenever the software driver sets an E bit in one or more receive descriptors, the driver follows that with a write to RDAR.

Chapter 37

IEEE 1588

37.1 Introduction

The objective of the IEEE1588 standard is to specify a protocol to synchronize independent clocks running on separate nodes of a distributed measurement and control system to a high degree of accuracy and precision. The clocks communicate with each other over a communication network. The protocol generates a master-slave relationship among the clocks in the system. Within a given subnet of a network, there will be a single master clock. All clocks ultimately derive their time from a clock known as the grandmaster clock.

The protocol will enable heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize. The protocol will support system wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications will be enhanced by having an accurate system wide sense of time achieved by having local clocks in each sensor, actuator, or other system device. Existing protocols for clock synchronization are not optimal for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with millisecond synchronization requirements.

Although IEEE1588 allows software-only implementations, hardware-assisted implementations deliver more precise clock synchronization.

The simplest IEEE1588 implementations include ordinary applications at the top of the network protocol stack and generate time stamps at the application level. Typically, this implementation incurs the largest protocol stack delay fluctuation, thus yielding the least accuracy as the largest amount of error is introduced into the time stamp.

Hardware-assisted methods achieve the greatest accuracy due to the fact that they generate time stamps as close to the wire as possible, at the physical layer. Network-protocol-delay fluctuations, for these implementations, typically range from sub-microseconds to nanoseconds.

In this block, the IEEE1588 implementation is a Hardware-assisted method. The hardware assist includes a time stamp unit to recognize PTP frames and transfer relevant time stamps, and a real time clock with high resolution to the sub-nanosecond accuracy.

37.2 IEEE1588 Block Diagram

Figure 487 shows the Time Stamp Unit (TSU) and the IEEE1588 Real Time Clock (RTC) modules integrated into the Engine.

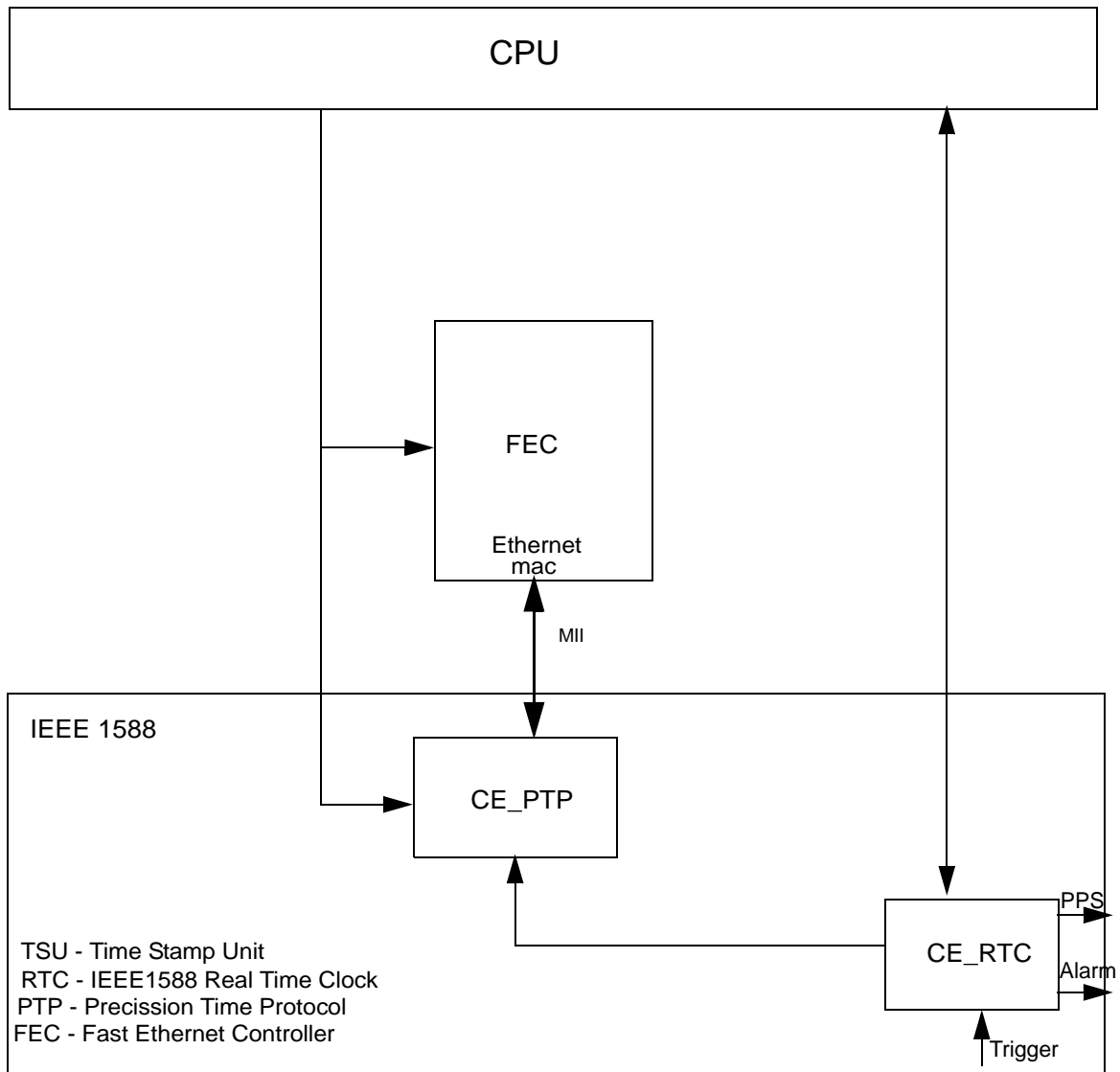


Figure 487. IEEE1588 block diagram

37.3 Time Stamp Unit (TSU) Key Features

- Support IEEE1588 V2
- Supports Ethernet systems
- Support IEEE1588 time stamping
- Recognition of PTP frames on receive side
 - Sync type
 - Delay_Req type

- Delay_Resp type
- Follow_Up type
- Management type
- Pdelay_Req
- Pdealy_Resp
- Announce
- Pdelay_Resp_Follow_Up
- Signaling
- Support PTP frame with VLAN tag (single VLAN)
- Support the detection of a new Ethernet type (IEEE1588 V2)
- Supports MII Physical I/F
- Support Ethernet frame monitoring on full and half duplex modes
- Supports slave and master mode
- Support single- or multi- IEEE1588 masters
- Support out-of-band mode of operation
 - Out-of-band (time stamp is read via host I/F)
- Time stamp captured close to the physical line
- Support the ability to timestamp Tx and Rx packets based on signals that are generated by the PHY upon detection of Start-Of-Frame appearing on the line
- Supports configured values for the PTP header fields
- Supports configured values for fields offset
- Supports interrupt notification due to following:
 - Rx PTP frame detection (for all the event messages)
 - Tx PTP frame transmission which was marked by the software as a PTP frame
 - Rx and Tx time stamp overrun error

37.4 IEEE1588 Real Time Clock (RTC) Key Features

- Support single IEEE1588 RTC.
- Support timer frequency compensation.
- Support timer offset update.
- Support configured source of clock.
- Time stamp capture on two general purpose external triggers through new GPIO connection.
 - Maskable interrupts on GPIO time stamp trigger
 - Programmable polarity for both output trigger signals
- Two 64-bit alarm (future time) registers that hold the value of the future time.
 - Maskable interrupts on alarm
 - Programmable polarity for both alarm (future time) output signals

- One 64-bit FIPER start register. Used to define the starting time of PPS signals generation
- Three programmable timer output pulse period phase aligned with IEEE1588 timer clock
 - Pulse Per Second (PPS) signals are aligned to frequency and phase
 - Maskable interrupts associated with each pulse
- Separate maskable timer interrupt event register
- Phase aligned adjustable (divide by N) clock output

37.5 IEEE1588 Implementation Assumptions

The IEEE1588 V2 standard makes several assumptions about the environment in which it operates. The following assumptions must be met to ensure correct operation of this implementation:

- The network must support multicast and unicast communication
- It must be possible to prevent multicast messages from propagating beyond a subnet
- Each clock implementing the protocol must meet the physical performance requirements, including oscillator frequency and time adjustment range.
- A clock's stated properties, including its stratum and identifier, must accurately describe the clock.

The following assumptions must be met to achieve optimal clock synchronization performance:

- A clock may contain asymmetric delays in its time stamp mechanism or protocol path. If these asymmetries are not negligible, they must be correctly accounted for.
- Network delay between master and slave on a subnet must be constant
- Boundary clocks must be used to synchronize across subnets
- The time stamps used in PTP are generated as close to the physical layer as practical for a given clock implementation. In cases where the most accurate time stamps can be generated only after a message has actually been transmitted, the actual value is communicated in the Follow_Up message from the master clock.
- The computing power of clocks implementing the protocol must be great enough, and the number of clocks per subnet must be small enough, to meet the standard constraints.
- The inherent stability of a clock's oscillator must be adequate. (The oscillator serving as the time base for a PTP clock has an accuracy and stability such that a second measured in the time base of the PTP clock is within +/- 0.01% of the SI second over the rated temperature range of the PTP clock)

37.6 Modes of Operation

- Out-of-band - This mode is determined in the TSMR register. All the time stamps are transferred through the host interface, using dedicated registers. The host reads the transmitter's time stamps after last indication in the buffer descriptor, and the receiver's time stamps when a PTP bit is set in the RxBD. The software is responsible for setting the PTP bit in the TxBD. In both receiver and transmitter the hardware can set an interrupt before accessing the time stamps.
- PTP frame parsing options - All of the relevant parsing values and fields can be configured using TSPDR register, retaining high flexibility.

- Pulse per second - The FIPER registers (TMR_FIPER) can be configured by software in order to generate the pulse per second outputs.
- Alarm mode - The TMR_ALARM register can be configured by software as an alarm (future time) value. When the IEEE1588 RTC reaches this value, an interrupt is generated to the core, and a dedicated output trigger signal will be asserted.
- Input trigger - The TMR_ETTS register configures a synchronous trigger to the IEEE1588 RTC in order to capture a time stamp.
- Source clock select - This mode is configured in the Timer Control Register (TMR_CTRL). There are two options for the source clock that can be connected to the IEEE1588 RTC.
- Time Stamp Point - The PTP_TSMR register can be configured to sample the IEEE1588 RTC according to SFD detection or according to an external time stamp lock trigger.

37.7 Memory Map/Register Definition

All IEEE1588 registers are 32-bits wide, located on 32-bit address boundaries, and should only be accessed as 32-bit quantities.

All addresses used in this chapter are offsets from RTC Base and PTP Base. (See “Memory map” chapter 2) [Table 498](#) lists the PTP Registers.

Table 498. PTP Registers

Internal Address	Register	R/W	Size
Time Stamp Unit Mode Registers			
0x0	PTP_TSPDR1	R/W	32bit
0x4	PTP_TSPDR2	R/W	32bit
0x8	PTP_TSPDR3	R/W	32bit
0xc	PTP_TSPDR4	R/W	32bit
0x10	PTP_TSPOV	R/W	32bit
0x14	PTP_TSMR	R/W	32bit
0x18	PTP_TMR_PEVENT	R/W	32bit
0x1c	PTP_TMR_PEMASK	R/W	32bit
0x20	TMR_RXTS_H	R	32bit
0x24	Reserved	R	32bit
0x28	Reserved	R	32bit
0x2c	Reserved	R	32bit
0x30	TMR_RXTS_L	R	32bit
0x34	Reserved	R	32bit
0x38	Reserved	R	32bit

Table 498. PTP Registers

Internal Address	Register	R/W	Size
Time Stamp Unit Mode Registers			
0x3c	Reserved	R	32bit
0x40	TMR_TXTS_H	R	32bit
0x44	Reserved	R	32bit
0x48	Reserved	R	32bit
0x4c	Reserved	R	32bit
0x50	TMR_TXTS_L	R	32bit
0x54	Reserved	R	32bit
0x58	Reserved	R	32bit
0x5c	Reserved	R	32bit
0x60	PTP_TSPDR5	R/W	32bit
0x64	PTP_TSPDR6	R/W	32bit
0x68	PTP_TSPDR7	R/W	32bit

Table 499 lists the RTC Registers.

Table 499. RTC Registers

Internal Address	Register	R/W	Size
IEEE1588 Timer Mode Registers			
0x0	TMR_CTRL	R/W	32bit
0x4	TMR_TEVENT	R/W	32bit
0x8	TMR_TEMASK	R/W	32bit
0xc	TMR_CNT_L	R/W	32bit
0x10	TMR_CNT_H	R/W	32bit
0x14	TMR_ADD	R/W	32bit
0x18	TMR_ACC	R	32bit
0x1c	TMR_PRSC	R/W	32bit
0x20	TMROFF_L	R/W	32bit
0x24	TMROFF_H	R/W	32bit
0x28	TMR_ALARM1_L	R/W	32bit
0x2c	TMR_ALARM1_H	R/W	32bit

Table 499. RTC Registers (continued)

Internal Address	Register	R/W	Size
IEEE1588 Timer Mode Registers			
0x30	TMR_ALARM2_L	R/W	32bit
0x34	TMR_ALARM2_H	R/W	32bit
0x38	TMR_FIPER1	R/W	32bit
0x3c	TMR_FIPER2	R/W	32bit
0x40	TMR_FIPER3	R/W	32bit
0x44	TMR_ETTS1_L	R	32bit
0x48	TMR_ETTS1_H	R	32bit
0x4c	TMR_ETTS2_L	R	32bit
0x50	TMR_ETTS2_H	R	32bit
0x54	TMR_FSV_L	R/W	32bit
0x58	TMR_FSV_H	R/W	32bit

NOTE

Access to PTP and RTC registers is not permitted if these are frozen by ME_PCTL61 and ME_PCTL62.

37.8 Time Stamp Unit Mode Registers

37.8.1 Time Stamp Unit Parsing Definitions Register 1 (PTP_TSPDR1)

The TSU Parsing Definition Register (PTP_TSPDR1) is shown in [Figure 488](#).

The PTP_TSPDR1 register is used to determine the values of the Ethernet type field and for UDP/IPv4 or UDP/IPv6 only, the IP type field.

The values for the parsing can be modified.

Note: The default values are according to IEEE1588 V1 [Table 506](#). For IEEE1588 V2 the relevant field should be configured according to table [Table 508](#).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ETT															
Reset	0x0800															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	IPT								RESERVED							
Reset	0x11															
R/W	R/W															
Addr	0x4880/4900															

Figure 488. PTP_TSPDR1 Register

[Table 500](#) describes the PTP_TSPDR1 fields.

Table 500. PTP_TSPDR1 Field Descriptions

Bits	Name	Description
0-15	ETT	Ethernet Type Field. This field indicates the protocol used in the data field of the frame For IPv4 or IPv6, this field should be configured to 0x0800 For PTP over IEEE 802.3/Ethernet, this field should be configured to 0x88F7 default value is 0x0800
16-23	IPT	IP Type Field. For PTP over UDP/IPv4 or PTP over UDP/IPv6, this field should be configured to 0x11 default value is 0x11
24-31	—	RESERVED

37.8.2 Time Stamp Unit Parsing Definitions Register 2(PTP_TSPDR2)

The TSU Parsing Definition Register 2 (PTP_TSPDR2) is shown in [Figure 489](#).

The PTP_TSPDR2 register is used to determine the values of the UDP port field for UDP/IPv4 or UDP/IPv6 only. For any other protocol, this register is ignored.

The values for the parsing can be modified.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DPNGE															
Reset	0x0140															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DPNEV															
Reset	0x013F															
R/W	R/W															
Addr	0x4884/4904															

Figure 489. PTP_TSPDR2 Register

Table 501 describes the PTP_TSPDR2 fields.

Table 501. PTP_TSPDR2 Field Descriptions

Bits	Name	Description
0-15	DPNGE	Destination port field value, represents PTP general For PTP over UDP/IPv4 or PTP over UDP/IPv6 the recommended value according to IEEE1588 standard (Table 506) is 0x0140. default value is 0x0140
16-31	DPNEV	Destination port field value, represents PTP event For PTP over UDP/IPv4 or PTP over UDP/IPv6 the recommended value according to IEEE1588 standard (Table 506) is 0x013F default value is 0x013F

37.8.3 Time Stamp Unit Parsing Definitions Register 3 (PTP_TSPDR3)

The TSU Parsing Definition Register 3 (PTP_TSPDR3) is shown in Figure 490.

The PTP_TSPDR3 register is used to determine the values of the PTP control field.

The values for the parsing can be modified and are used only when PTP_TSMR[PTPV] = 0. When PTP_TSMR[PTPV] = 1, this register is ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SYCTL								DRCTL							
Reset	0x00								0x01							
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DRPCTL								FUCTL							
Reset	0x03								0x02							
R/W	R/W															
Addr	0x4888/4908															

Figure 490. PTP_TSPDR3 Register Table 502 describes the PTP_TSPDR3 fields.

Table 502. PTP_TSPDR3 Field Descriptions

Bits	Name	Description
0-7	SYCTL	Control field value represents Sync message. The recommended value according to IEEE1588 standard (Table 506) is 0x00. default value is 0x00
8-15	DRCTL	Control field value represents Delay_Req message. The recommended value according to IEEE1588 standard (Table 506) is 0x01. default value is 0x01
16-23	DRPCTL	Control field value represents Delay_Resp message. The recommended value according to IEEE1588 standard (Table 506) is 0x03. default value is 0x03
24-31	FUCTL	Control field value represents Follow_Up message. The recommended value according to IEEE1588 standard (Table 506) is 0x02. default value is 0x02

37.8.4 Time Stamp Unit Parsing Definitions Register 4 (PTP_TSPDR4)

The TSU Parsing Definition Register 4 (PTP_TSPDR4) is shown in Figure 491.

The PTP_TSPDR4[MACTL] field is used to determine the values of the PTP control field and is in continuance to the PTP_TSPDR3 register. The values for the parsing can be modified and are used only when PTP_TSMR[PTPV] = 0. When PTP_TSMR[PTPV] = 1, this field is ignored. In addition, PTP_TSPDR4 register provides a configured value to detect a VLAN field. This value can be detected during bytes 12-13 and cause a shifting of 4 bytes for all other parsing offsets. In case of a VLAN frame, the offsets should be configured exactly as for non VLAN frame. This field is relevant for both PTP_TSMR[PTPV] = 0 and PTP_TSMR[PTPV] = 1.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MACTL								RESERVED							
Reset	0x04															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	VLAN															
Reset	0x8100															
R/W	R/W															
Addr	0x488c/490c															

Figure 491. PTP_TSPDR4 Register

Table 503 describes the PTP_TSPDR4 fields.

Table 503. PTP_TSPDR4 Field Descriptions

Bits	Name	Description
0-7	MACTL	Control field value represents Management message. The recommended value according to IEEE1588 standard (Table 506) is 0x04. default value is 0x04
8-15	—	RESERVED
16-31	VLAN	VLAN tag value (0x8100) default value is 0x8100

37.8.5 Time Stamp Unit Parsing Definitions Register 5 (PTP_TSPDR5)

The TSU Parsing Definition Register 5 (PTP_TSPDR5) is shown in Figure 492.

The PTP_TSPDR5 register is used to determine the values of the PTP MessageType field.

The values for the parsing can be modified and are used only when PTP_TSMR[PTPV] = 1. When PTP_TSMR[PTPV] = 0, this register is ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SYMSG				RESERVED				DRMSG				RESERVED			
Reset	0x0				0x0				0x1				0x0			
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	PDRMSG				RESERVED				PDRPMSG				RESERVED			
Reset	0x2				0x0				0x3				0x0			
R/W	R/W															
Addr	0x488c/490c															

Figure 492. PTP_TSPDR5 Register

Table 504 describes the PTP_TSPDR5 fields.

Table 504. PTP_TSPDR5 Register

Bits	Name	Description
0-3	SYMSG	Message Type field value represents Sync message. The recommended value according to IEEE1588 standard (Table 508) 0x0. default value is 0x0
4-7	—	RESERVED set to 0x0
8-11	DRMSG	Message Type field value represents Delay_Req message. The recommended value according to IEEE1588 standard (Table 508) 0x1. default value is 0x1
12-15	—	RESERVED set to 0x0
16-19	PDRMSG	Message Type field value represents Pdelay_Req message. The recommended value according to IEEE1588 standard (Table 508) 0x2. default value is 0x2
20-23	—	RESERVED set to 0x0
24-27	PDRPMSG	Message Type field value represents Pdelay_Resp message. The recommended value according to IEEE1588 standard (Table 508) 0x3. default value is 0x3
28-31	—	RESERVED set to 0x0

37.8.6 Time Stamp Unit Parsing Definitions Register 6 (PTP_TSPDR6)

The TSU Parsing Definition Register 6 (PTP_TSPDR6) is shown in Figure 493.

The PTP_TSPDR6 register is used to determine the values of the PTP Message Type field and is in continuance to the PTP_TSPDR5 register.

The values for the parsing can be modified and are used only when PTP_TSMR[PTPV] = 1. When PTP_TSMR[PTPV] = 0, this register is ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FUMSG				RESERVED				DRPMSG				RESERVED			
Reset	0x8				0x0				0x9				0x0			
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	PDFUMSG				RESERVED				ANNMSG				RESERVED			
Reset	0xA				0x0				0xB				0x0			
R/W	R/W															
Addr	0x48E0/4960															

Figure 493. PTP_TSPDR6 Register

Bits	Name	Description
0-3	FUMSG	Message Type field value represents Follow_Up message. The recommended value according to IEEE1588 standard (Table 508) is 0x8. default value is 0x8
4-7	—	RESERVED set to 0x0
8-11	DRPMSG	Message Type field value represents Delay_Resp message. The recommended value according to IEEE1588 standard (Table 508) is 0x9. default value is 0x9
12-15	—	RESERVED set to 0x0
16-19	PDFUMSG	Message Type field value represents Pdelay_Resp_Follow_Up message. The recommended value according to IEEE1588 standard (Table 508) is 0xA. default value is 0xA
20-23	—	RESERVED set to 0x0
24-27	ANNMSG	Message Type field value represents Announce message. The recommended value according to IEEE1588 standard (Table 508) is 0xB. default value is 0xB
28-31	—	RESERVED set to 0x0

37.8.7 Time Stamp Unit Parsing Definitions Register 7 (PTP_TSPDR7)

The TSU Parsing Definition Register 7 (PTP_TSPDR7) is shown in [Figure 494](#).

The PTP_TSPDR7 register is used to determine the values of the PTP MessageType field and is in continuance to the PTP_TSPDR6 register.

The values for the parsing can be modified and are used only when PTP_TSMR[PTPV] = 1. When PTP_TSMR[PTPV] = 0, this register is ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SIGMSG				RESERVED				MAMSG				RESERVED			
Reset	0xC				0x0				0xD				0x0			
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TRANSPEC				RESERVED											
Reset	0x0															
R/W	R/W															
Addr	0x48E4/4964															

Figure 494. PTP_TSPDR7 Register

[Table 505](#) describes the PTP_TSPDR7 fields.

Table 505. PTP_TSPDR7 Field Descriptions

Bits	Name	Description
0-3	SIGMSG	Message Type field value represents Signaling message. The recommended value according to IEEE1588 standard (Table 508) is 0xC. default value is 0xC
4-7	—	RESERVED set to 0x0
8-11	MAMSG	Message Type field value represents Management message. The recommended value according to IEEE1588 standard (Table 508) is 0xD. default value is 0xD
12-15	—	RESERVED set to 0x0
16-19	TRANSPEC	transportSpecific field value represents a subtype of the Ethertype. Note: This field is parsed only when PTP_TSMR[ENTRNR]=1, PTP_TSMR[PTPV] = 1, and PTP_TSMR[ENUUDP] = 0. default value is 0x0
20-31	—	RESERVED set to 0x0

37.8.8 Time Stamp Unit Parsing Offset Values (PTP_TSPOV)

The TSU Parsing Offset Values (PTP_TSPOV) is shown in [Figure 495](#).

In order to detect a PTP frame, the TSU should parse several fields in the Ethernet frame. TSU offset values register (PTP_TSPOV) is used to determine the offset of each field in the Ethernet frame. In case the structure of the frame has been changed or manipulated the fields offsets can be configured.

[Table 506](#) describes the structure of the PTP frame according to IEEE1588 V1 (PTP_TSMR[PTPV]=0).

Table 506. PTP Frame Structure according to IEEE1588 V1

Layer Name	Octet Number	Field Name	Field Value
IP	12-13	Ethernet Type Field	0x0800
UDP	23	IP Type Field.	0x11
PTP Event	36-37	UDP port	0x013F
PTP General	36-37	UDP port	0x0140
Sync	74	Control	0x00
Delay_Req	74	Control	0x01
Follow_Up	74	Control	0x02
Delay_Resp	74	Control	0x03
Management	74	Control	0x04

[Table 508](#) describes the structure of the PTP frame according to IEEE1588 V2 (PTP_TSMR[PTPV]=1) when working with PTP over UDP/IPv4 or PTP over UDP/IPv6. Note: X signifies the PTP common message header offset.

Table 507. PTP Frame Structure according to IEEE1588 V2 PTP over UDP/IPv4 or PTP over UDP/IPv6

Layer Name	Octet Number	Field Name	Field Value according to standard
IP	12-13	Ethernet Type Field	0x0800
UDP	23	IP Type Field.	0x11
PTP Event	36-37	UDP port	0x013F
PTP General	36-37	UDP port	0x0140
Sync	X	Message Type	0x0
Delay_Req	X	Message Type	0x1
Pdelay_Req	X	Message Type	0x2
Pdelay_Resp	X	Message Type	0x3
Follow_Up	X	Message Type	0x8
Delay_Resp	X	Message Type	0x9
Pdelay_Resp_Follow_Up	X	Message Type	0xA

Layer Name	Octet Number	Field Name	Field Value according to standard
Announce	X	Message Type	0xB
Signaling	X	Message Type	0xC
Management	X	Message Type	0xD

Table 508 describes the structure of the PTP frame according to IEEE1588 V2 (PTP_TSMR[PTPV]=1) when working with PTP over IEEE 802.3/Ethernet. Note: X signifies the PTP common message header offset.

Table 508. PTP Frame Structure according to IEEE1588 V2 PTP over IEEE 802.3/Ethernet

Layer Name	Octet Number	Field Name	Field Value according to standard
EtherType	12-13	Ethernet Type Field	0x88F7
Sync	X	Message Type	0x0
Delay_Req	X	Message Type	0x1
Pdelay_Req	X	Message Type	0x2
Pdelay_Resp	X	Message Type	0x3
Follow_Up	X	Message Type	0x8
Delay_Resp	X	Message Type	0x9
Pdelay_Resp_Follow_Up	X	Message Type	0xA
Announce	X	Message Type	0xB
Signaling	X	Message Type	0xC
Management	X	Message Type	0xD

The parser allows to detect a PTP header as defined in IEEE1588 V1 and V2. In order to retain high flexibility, all of the parsing values and their offsets can be configured.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ETTOF								IPTOF							
Reset	0x0C								0x17							
R/W																
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	UDOF								PTOF							
Reset	0x24								0x4A							
R/W	R/W															
Addr	0x4890/4910															

Figure 495. PTP_TSPOV Register

Table 509 describes the PTP_TSPOV fields.

Table 509. PTP_TSPOV Field Descriptions

Bits	Name	Description
0-7	ETTOF	Ethernet type offset (0x0C) default value is 0x0C
8-15	IPTOF	IP type offset (0x17) Note: When PTP_TSMR[ENUUDP] = 0 this field is ignored. default value is 0x17
16-23	UDOF	UDP port offset (0x24) Note: When PTP_TSMR[ENUUDP] = 0 this field is ignored. default value is 0x24
24-31	PTOF	Control field offset (when PTP_TSMR[PTPV]=0) (0x4A) MessageType field offset (when PTP_TSMR[PTPV]=1) default value is 0x4A

37.8.9 Time Stamp Unit Mode Register (PTP_TSMR)

The TSU Mode Register (PTP_TSMR) is shown in Figure 496.

The PTP_TSMR register is used to determine the modes of operation of the TSUs.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Field	RESERVED											OPM ODE1	OPM ODE2	OPM ODE3	OPM ODE4		
Reset													0	0	0	0	
R/W	R/W																
Addr																	
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Field	RESERVED							PTPV	ENUUDP	ENT RN	RESERVED			EN1	EN2	EN3	EN4
Reset								0	0	0			0	0	0	0	
R/W	R/W																
Addr	0x4894/4914																

Figure 496. PTP_TSMR Register

Table 510 describes the PTP_TSMR fields.

Table 510. PTP_TSMR Field Descriptions

Bits	Name	Description
0-11	—	RESERVED
12	OPMODE1	TSU mode of operation. 0 - out-of-band 1 - Reserved Note: if OPMODE is set, the Ethernet receive soft preamble mode should be set as well (MACCFG2[SRP]) default value is 0
13	OPMODE2	RESERVED
14	OPMODE3	RESERVED
15	OPMODE4	RESERVED
16-22	—	RESERVED
23	PTPV	IEEE1588 version of operation 0 - IEEE1588 V1 1 - IEEE1588 V2 default value is 0
24	ENUUDP	Enable UDP 0 - UDP disabled 1 - UDP enabled Note: This field is valid only when PTPV= 1 default value is 0
25	ENTRN	Enable transportSpecific 0 - Disable parsing of the transportSpecific field in the common message header 1 - Enable parsing of the transportSpecific field in the common message header Note: This field is valid only when PTPV= 1 and ENUUDP= 0 default value is 0
26-27	—	RESERVED
28	EN1	0 - Time Stamp Unit Disabled 1 - Time Stamp Unit Enabled default value is 0
29	EN2	RESERVED
30	EN3	RESERVED
31	EN4	RESERVED

37.8.10 Timer PTP Event Register (PTP_TMR_PEVENT)/ Timer PTP Mask Register (PTP_TMR_PEMASK)

The TSU Event Register (PTP_TMR_PEVENT) and Mask Register are shown in [Figure 497](#).

The PTP_TMR_PEVENT is used as the TSU event register. The PTP_TMR_PEVENT reports events recognized on the Ethernet channel and generates interrupts. After event recognition, the TSU sets the

corresponding PTP_TMR_PEVENT bit. The PTP_TMR_PEVENT bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the CPU clears the internal interrupt request.

Interrupts generated by the TSU event register (PTP_TMR_PEVENT) can be masked in the TSU mask register (PTP_TMR_PEMASK), which has the same bit format as PTP_TMR_PEVENT. If a PTP_TMR_PEMASK bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	EXR	OVR1	OVT1	SYRE1	DRQRE1	TXE1	PDRQRE1	PDRSRE1	EXT	OVR2	OVT2	SYRE2	DRQRE2	TXE2	PDRQRE2	PDRSRE2
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	OVR3	OVT3	SYRE3	DRQRE3	TXE3	PDRQRE3	PDRSRE3	—	OVR4	OVT4	SYRE4	DRQRE4	TXE4	PDRQRE4	PDRSRE4
Reset		0	0	0	0	0	0	0		0	0	0	0	0	0	0
R/W	R/W															
Addr	0x4898/4918 0x489c/491c															

Figure 497. Event Register (TMR_PEVENT) / Mask Register (TMR_PEMASK)

Table 511 describes the TMR_PEVENT/TMR_PEMASK fields.

Table 511. TMR_PEVENT/TMR_PEMASK Field Descriptions

Bits	Name	Description
0	EXR	External trigger was not received to receiver from PHY. No time stamp taken default value is 0
1	OVR1	Overflow occurred in receiver time stamp register. default value is 0
2	OVT1	Overflow occurred in transmitter time stamp register. default value is 0
3	SYRE1	Sync frame has been received. default value is 0
4	DRQRE1	Delay_req frame has been received. default value is 0

Table 511. TMR_PEVENT/TMR_PEMASK Field Descriptions

Bits	Name	Description
5	TXE1	PTP frame has been transmitted. default value is 0
6	PDRQRE1	Pdelay_Req frame has been received. default value is 0
7	PDRSRE1	Pdelay_Resp frame has been received. default value is 0
8	EXT	External trigger was not received to transmitter from PHY. No time stamp taken default value is 0
9	OVR2	RESERVED
10	OVT2	RESERVED
11	SYRE2	RESERVED
12	DRQRE2	RESERVED
13	TXE2	RESERVED
14	PDRQRE2	RESERVED
15	PDRSRE2	RESERVED
16	—	RESERVED
17	OVR3	RESERVED
18	OVT3	RESERVED
19	SYRE3	RESERVED
20	DRQRE3	RESERVED
21	TXE3	RESERVED
22	PDRQRE3	RESERVED
23	PDRSRE3	RESERVED
24	—	RESERVED
25	OVR4	RESERVED
26	OVT4	RESERVED
27	SYRE4	RESERVED
28	DRQRE4	RESERVED
29	TXE4	RESERVED
30	PDRQRE4	RESERVED
31	PDRSRE4	RESERVED

37.8.11 Time Stamp Unit Receiver Time High (TMR_UC_RXTS_H)/Time Stamp Unit Receiver Time Low (TMR_UC_RXTS_L)/Time Stamp Unit Transmitter Time High (TMR_UC_TXTS_H)/Time Stamp Unit Transmitter Time Low (TMR_UC_TXTS_L)

The TMR_UC_TXTS_L/TMR_UC_TXTS_H and TMR_UC_RXTS_L/TMR_UC_RXTS_H registers are shown in [Figure 498](#).

The TMR_UC_TXTS_L/TMR_UC_TXTS_H registers are used to capture the time stamp of a recognized PTP frame on the transmitter lines. The TSU samples the IEEE1588 RTC 64-bit counter immediately after detecting the message time stamp point (See MTSP bit in TSMR_CTRL). The sampled value becomes valid only after one of the frame events. In this case, the CPU reads the value in order to calculate the synchronizations commands.

TMR_UC_RXTS_L/TMR_UC_RXTS_H have the same purpose on the receiver lines.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TSVAL															
Reset	0000_0000_0000_0000															
R/W	R															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TSVAL															
Reset	0000_0000_0000_0000															
R/W	R															
Addr	UC_RXTS_H: 0x48a0/48a4/48a8/48ac/4920/4924/4928/492c UC_RXTS_L: 0x48b0/48b4/48b8/48bc/4930/4934/4938/493c UC_TXTS_H: 0x48c0/48c4/48c8/48cc/4940/4944/4948/494c UC_TXTS_L: 0x48d0/48d4/48d8/48dc/4950/4954/4958/495c															

Figure 498. TMR_UC_RXTS_H, TMR_UC_RXTS_L, TMR_UC_TXTS_H, TMR_UC_TXTS_L Registers

[Table 512](#) describes the fields.

Table 512. TMR_UC_TXTS_L, TMR_UC_RXTS_L, TMR_UC_TXTS_H, TMR_UC_RXTS_H Field Descriptions

Bits	Name	Description
0-31	TSVAL	Time stamp value of the PTP packet

37.9 IEEE1588 Timer Mode Registers

37.9.1 Timer Control Register (TMR_CTRL)

The TMR_CTRL defines control fields relevant to the IEEE1588 timer. TMR_CTRL is a writable register in order to reset, configure, and initialize the IEEE1588 timer clock. [Figure 499](#) describes the definition for the TMR_CTRL register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ALM1P	ALM2P	—	—	PPSW	DLPB1	TCLK_PERIOD									
Reset	0	0			0		0000_0000_0000									
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DLPB2	RESERVED					ETEP2	ETEP1	COPH	CIPH	TMSR	DBG	BYP	TE	CKSEL	
Reset							0	0	0	0	0	0	0	0	00	
R/W	R/W															
Addr	0x4800															

Figure 499. TMR_CTRL Register

[Table 513](#) describes the TMR_CTRL fields.

Table 513. TMR_CTRL Field Descriptions

Bits	Name	Description
0	ALM1P	Alarm1 output polarity 0 active high output 1 active low output default value is 0
1	ALM2P	Alarm2 output polarity 0 active high output 1 active low output default value is 0
2	—	RESERVED
3	—	RESERVED

Table 513. TMR_CTRL Field Descriptions

Bits	Name	Description
4	PPSW	Pulse Per Second width 0 PPS signal width is identical to the width selected timer clock 1 PPS signal width is identical to the width of the IEEE1588 timer reference output clock Note: When TMR_PRSC[PRSC_OCK] is set to 16'h1 this bit has no meaning default value is 0
5	DLPB1	Diagnostic Loopback mode. When set, PPS1 signal is directed to external trigger 1 for debugging purpose. Default value is 0. 0 Diagnostic mode is disabled for PPS1 1 Diagnostic mode is enabled for PPS1
6-15	TCLK_PERIOD	IEEE1588 timer reference clock period. The timer clock counter will increment by TCLK_PERIOD every time the accumulator register overflows. This clock period must be larger than the clock period of the timer reference clock. This field can be programmed to 10'h1 to count clock ticks for applications that do not desire to add the clock period. default value is 10'h0 Note: It is recommended to work with clock ticks when the oscillator period can not be represented by an integer number
16	DLPB2	Diagnostic Loopback mode. When set, PPS1 signal is directed to external trigger 1 for debugging purpose. Default value is 0. 0 Diagnostic mode is disabled for PPS1 1 Diagnostic mode is enabled for PPS1
17-21	—	RESERVED
22	ETEP2	External trigger 2 edge polarity 0 time stamp on the rising edge of the external trigger 1 time stamp on the falling edge of the external trigger default value is 0 Note: When DLPB2 is set to 1, this bit is not relevant
23	ETEP1	External trigger 1 edge polarity 0 time stamp on the rising edge of the external trigger 1 time stamp on the falling edge of the external trigger default value is 0 Note: When DLPB1 is set to 1, this bit is not relevant
24	COPH	Generated clock (prescale) output phase 0 non-inverted divided clock is output 1 inverted divided clock is output default value is 0
25	CIPH	External oscillator input clock phase 0 non-inverted frequency tuned timer input clock 1 inverted frequency tuned timer input clock default value is 0

Table 513. TMR_CTRL Field Descriptions

Bits	Name	Description
26	TMSR	Timer soft reset. When enabled, it resets all the timer registers and state machines. 0 normal operation 1 places entire timer in reset except control and configuration registers default value is 0
27	DBG	Debug mode. When set, allows writing to the read only timer registers 0 normal operation 1 enable debug mode default value is 0
28	BYP	Bypass drift compensated clock 0 64-bit clock counter is incremented on the accumulator overflow (both phase and frequency are fixed) 1 64-bit clock counter is directly driven from the external oscillator ignoring accumulator overflow (only phase is fixed) default value is 0
29	TE	IEEE1588 Timer enable. If not enabled, all the timer registers and state machines are disabled. 0 timer not enabled 1 timer enabled and resumes normal operation default value is 0
30-31	CKSEL	Selection of the source clock for the IEEE1588 timer clock 00 external clock source 01 IPG clock 1x Reserved for block

37.9.2 Timer Event Register (TMR_TEVENT)/Timer Event Mask Register (TMR_TEMASK)

The IEEE1588 timer implementation requires several interrupt event signals. The needs to add a new interrupt output line for the timer independent of the existing Tx, Rx and Error interrupts. The IEEE1588 timer interrupt does not require any interrupt coalescing. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the event mask register (EMASK), the event also causes a hardware interrupt at the Programmable Interrupt Controller (PIC). A bit in the timer event register is cleared by writing a 1 to that bit position.

Figure 500 shows the TMR_EVENT register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RESERVED					ETS2	ETS1	RESERVED					ALM1	ALM2		
Reset						0	0						0	0		
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	FS	RESERVED						PP1	PP2	PP3	RESERVED					
Reset	0							0	0	0						
R/W	R/W															
Addr	0x4804/4808															

Figure 500. Event Register (TMR_TEVENT) / Mask Register (TMR_TEMASK)

Table 514 describes the fields.

Table 514. TMR_TEVENT/TMR_TEMASK Field Descriptions

Bits	Name	Description
0-5	—	RESERVED
6	ETS2	External trigger 2 time stamp sampled 0 external trigger time stamp not sampled 1 external trigger time stamp sampled default value is 0
7	ETS1	External trigger 1 time stamp sampled 0 external trigger time stamp not sampled 1 external trigger time stamp sampled default value is 0
8-13	—	RESERVED
14	ALM1	Current time equaled alarm 1 time register 0 alarm time has not be reached yet 1 alarm time has been reached default value is 0
15	ALM2	Current time equaled alarm 2 time register 0 alarm time has not been reached yet 1 alarm time has been reached default value is 0
16	FS	Current time equaled the FIPER Start Value (FSV) 0 FIPER value has not been reached yet 1 FIPER value has been reached default value is 0
17-23	—	RESERVED

Table 514. TMR_TEVENT/TMR_TEMASK Field Descriptions

Bits	Name	Description
24	PP1	Indicates that a periodic pulse has been generated based on FIPER1 register. 0 periodic pulse not generated 1 periodic pulse generated default value is 0
25	PP2	Indicates that a periodic pulse has been generated based on FIPER2 register. 0 periodic pulse not generated 1 periodic pulse generated default value is 0
26	PP3	Indicates that a periodic pulse has been generated based on FIPER3 register. 0 periodic pulse not generated 1 periodic pulse generated default value is 0
27-31	—	RESERVED

37.9.3 Timer Counter Register (TMR_CNT_L/TMR_CNT_H)

The TMR_CNT_L/ TMR_CNT_H registers are shown in [Figure 501](#).

The timer counter register (TMR_CNT_L/TMR_CNT_H) represents accurate time in terms of clock ticks or in nano-seconds. This is a read/write register. Writing to these registers will override the previous time.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMR_CNT															
Reset	0000_0000_0000_0000															
R/W	W/R															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TMR_CNT															
Reset	0000_0000_0000_0000															
R/W	W/R															
Addr	0x480c/4810															

Figure 501. TMR_CNT_L/TMR_CNT_H Register

Table 515 describes the fields.

Table 515. TMR_CNT_L/TMR_CNT_H Field Descriptions

Bits	Name	Description
0-31	TMR_CNT	<p>H/L value of the current time counter. Current time is calculated by adding TMROFF_H/L with the TMR_CNT_H/L counter. This register can be written through the register writes. Writes to the TMR_CNT_L register copies the written value into the shadow TMR_CNT_L register. Writes to the TMR_CNT_H register copies the values written into the shadow TMR_CNT_H register. Contents of the shadow registers are copied into the TMR_CNT_L and TMR_CNT_H registers following a write into the TMR_CNT_H register. Writes to these registers have precedence over the timer increment. The TMR_CNT_L register must be written first.</p> <p>Reads from the TMR_CNT_L register copies the entire 64-bit clock time of the read enable into the TMR_CNT_H/L shadow registers. Read instruction from the TMR_CNT_H register reads the value stored in the TMR_CNT_H shadow register. In order to get the correct 64-bit TMR_CNT_H/L counter values, the TMR_CNT_L register must be read first.</p>

37.9.4 Timer Addend Register (TMR_ADD)

The Timer Drift Compensation Addend Register (TMR_ADD) is used to hold the timer frequency compensation value (FreqCompensationValue). The nominal frequency of the clock counter is determined by the Frequency division ratio (FreqDivRatio) and the clock frequency (FreqClock). This register is programmed with $2^{32}/\text{FreqDivRatio}$. Frequency division ratio (FreqDivRatio) is the ratio between the frequency of the oscillator (TimerOsc) and the desired clock frequency (NominalFreq). FreqDivRatio is a design constant chosen to be greater than 1.0001. The ADDEND value is added to the 32-bit accumulator register at every rising edge of the oscillator clock (TimerOsc). The clock counter is incremented at every carry pulse of the accumulator.

Figure 502 shows the TMR_ADD register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ADDEND															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	ADDEND															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4814															

Figure 502. TMR_ADD Register

Table 516 describes the TMR_ADD fields.

Table 516. TMR_ADD Field Descriptions

Bits	Name	Description
0-31	ADDEND	Timer drift compensation addend register value. It is programmed with a value of $2^{32}/\text{FreqDivRatio}$. For example: FreqOsc = 50 MHz FreqClock = 40 MHz FreqDivRatio = 1.25 $\text{RTCAD} = \text{ceil}(2^{32}/1.25) = 0xCCCC_CCCD$ default value is 32'h0

37.9.5 Timer Accumulator Register (TMR_ACC)

The Timer Accumulator Register accumulates the value of the addend register into it. An overflow pulse of the accumulator is used to increment the timer clock (TMR_CNT) by TMR_CTRL[TCLK_PERIOD]. This register is read only. This register is writable only in debug mode (TMR_CTRL[DBG]).

Figure 503 shows the TMR_ACC register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TNR_ACC															
Reset	0000_0000_0000_0000															
R/W	R															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TMR_ACC															
Reset	0000_0000_0000_0000															
R/W	R															
Addr	0x4818															

Figure 503. TMR_ACC Register

Table 517 describes the TMR_ACC fields.

Table 517. TMR_ACC Field Descriptions

Bits	Name	Description
0-31	TMR_ACC	32bit timer accumulator value

37.9.6 Timer Prescale Register (TMR_PRSC)

The Timer Prescale Register is used to adjust output clock frequency. Figure 504 shows the TMR_PRSC register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RESERVED															
Reset																
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	PRSC_OCK															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x481c															

Figure 504. Prescale Register (TMR_PRSC)

Table 518 describes the TMR_PRSC fields.

Table 518. Prescale Register (TMR_PRSC)

Bits	Name	Description
0-15	—	RESERVED
16-31	PRSC_OCK	Output clock division/prescale factor. Output clock is generated by dividing the timer clock by this number. Setting the prescaler value to 16'h1 will generate an output clock in the same frequency as the selected timer clock. default value is 16'h0 ¹

¹ Setting the prescaler value to 16'h1 or 16'h0 will generate an output clock in the same frequency as the selected timer clock.

37.9.7 Timer Offset Register (TMROFF_L/TMROFF_H)

The TMROFF_L/TMROFF_H register is shown in Figure 505.

The timer offset register is used to update the timer to the current time by adding its value to the clock counter.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMROFF															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TMROFF															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4820/4824															

Figure 505. TMROFF_L/TMROFF_H Register

Table 519 describes the fields.

Table 519. TMROFF_L/TMROFF_H Field Descriptions

Bits	Name	Description
0-31	TMROFF	The TMROFF value, which is determined by the software calculations, updates the timer clock counter (TMR_CNT) with its value. TMROFF_L is written first and the update is executed. The update is executed when the hardware detects write transaction to the offset high register. default value is 32'h0

37.9.8 Alarm Time Register (TMR_ALARM_L/TMR_ALARM_H)

The TMR_ALARM_L/TMR_ALARM_H register is shown in [Figure 506](#).

The Alarm (future time) Comparator Register (TMR_ALARM_H/TMR_ALARM_L) holds alarm time for comparison with the current timer clock counter (TMR_CNT).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMR_ALARMn															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TMR_ALARMn															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4828/482c/4830/4834															

Figure 506. TMR_ALARMn register structure

[Table 520](#) describes the TMR_ALARMn fields.

Table 520. TMR_ALARMn Bit Setting

Bits	Name	Description
0-31	TMR_ALARMn	The IEEE1588 RTC interrupt is generated when the value of the RTC counter (TMR_CNT_L/TMR_CNT_H) is greater than or equal to TMR_ALARMn[TMR_ALARM]. This value should be configured before enabling the timer. default value is 32'h0

37.9.9 Timer Fixed Interval Period Register (TMR_FIPERn)

The Timer Fixed Interval Period Pulse Generator Register is used to generate periodic pulses. This register is reset with 0xFFFF_FFFF to prevent any false pulse upon initialization. The down count register loads the value programmed in the Fixed Period Interval Register (FIPER). The FIPER must be programmed before the timer is enabled and before the FIPER start value (FSV), which triggers the start of the down counting, is configured. At every tick of the FreqOsc, the counter decrements by the value of TMR_CTRL[TCLK_PERIOD]. It generates a pulse when the down counter value reaches zero regardless of the timer value. It reloads the down counter in the cycle following a pulse. The software is responsible for loading these registers with desired period intervals. There are three FIPER registers.

[Figure 507](#) shows the TMR_FIPER register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FIPER															
Reset	FFFF_FFFF_FFFF_FFFF															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	FIPER															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4838/483c/4840															

Figure 507. TMR_FIPERn Register

Table 521 describes the TMR_FIPERn fields.

Table 521. TMR_FIPERn Field Descriptions

Bits	Name	Description
0-31	FIPER ¹	<p>The FIPER value represents the rate of pulses generated at the output.</p> <p>The PPS signal is set with respect to the prescale clock. When the prescale value is different than one, the conditions below must be met:</p> <ol style="list-style-type: none"> In order to allow PPS alignment to phase and frequency, the following equations must result in an integer value: $\frac{\text{FIPER_VALUE} + \text{TMR_CTRL}[\text{TCLK_PERIOD}]}{\text{TMR_PRSC}[\text{PRSC_OCK}] * \text{TMR_CTRL}[\text{TCLK_PERIOD}]}$ $\text{FIPER_VALUE} > \text{TMR_CTRL}[\text{TCLK_PERIOD}] * \text{TMR_PRSC}[\text{PRSC_OCK}]$ The FIPER start value must be configured to a value which is $3 * \text{TMR_CTRL}[\text{TCLK_PERIOD}]$ less than the targeted value Assuming PPS should be set every 1 sec. (10^9 nsec), timer frequency is 100MHZ, and the $\text{TMR_CTRL}[\text{TCLK_PERIOD}]$ is 10, then the configured value should be: $\{10^9 / \text{TMR_CTRL}[\text{TCLK_PERIOD}]\}$hex - $\{\text{TMR_CTRL}[\text{TCLK_PERIOD}]\}$hex = 5F5E100- $\{\text{TMR_CTRL}[\text{TCLK_PERIOD}]\}$hex = 5F5E0F6 In order to keep tracking the prescale output clock, each time before enabling the FIPER, the FIPER must be reset by writing a new value to the register. <p>default value is 0</p>

¹ The FIPER must be programmed before the timer is enabled and before the FIPER start value (FSV), which triggers the start of the down counting, is configured. To disable the FIPER, put the maximum value in the FSV register. This way the trigger for the down counting will not be reached.

37.9.10 FIPER Start Register (TMR_FSV_L/TMR_FSV_H)

The FIPER Start Register Value indicates to the three FIPERs when to start the down counting.

The TMR_FSV_L/TMR_FSV_H register is shown in [Figure 506](#).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMR_FSV															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TMR_FSV															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4854/4858															

Figure 508. TMR_FSV_L/TMR_FSV_H register structure

Table 522. TMR_FSV_L/TMR_FSV_H Bit Setting

Bits	Name	Description
0-31	TMR_FSV	The FIPER Start value is used as an indicator to the FIPER when to start the down counting. default value is 32'h0

37.9.11 External Trigger Time Stamp Register (TMR_ETTS_L/TMR_ETTS_H)

The general purpose External Trigger Time Stamp Register (TMR_ETTS_H/L) holds the time stamp at the programmable edge of the external trigger. There are only two of these registers.

[Figure 509](#) shows the TMR_ETTS_H/TMR_ETTS_L register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ETTS															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	ETTS															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4844/4848/484c/4850															

Figure 509. TMR_ETTS_L/TMR_ETTS_H Register

Table 523 describes the fields.

Table 523. TMR_ETTS_L/TMR_ETTS_H Field Descriptions

Bits	Name	Description
0-31	ETTS	Time stamp value at the programmable edge of the external trigger. default value is 32'h0 ¹

¹ When the timer reaches the value configured in this register, a timestamp is sampled.

37.10 Time Stamp Unit (TSU)

The TSU logic implementation supports out-of-band mode of operation. As mentioned in the preceding sections, the hardware assumption is that the IEEE1588 is running on an Ethernet system.

In out-of-band method, the reporting is done using dedicated registers per interface. The software will read the transmitter’s time stamps after the “last” indication of the TxBD, and the receiver’s time stamps when a PTP bit was set in the RxBD. If there is sufficient time to guarantee that the software will be able to read the time stamp register between PTP frames, then the out-of-band method is sufficient.

The TSU logic supports MII Physical I/F.

In order to snapshot a precise time stamp, the hardware either observes the ports on the interfaces between the Media Access Controller (MAC) devices and the physical layer devices (PHY) to detect an SFD or receives an external lock trigger and captures the timestamp on the transmit and receive sides.

Figure 510 shows the time stamp point of a PTP frame with SFD detection.

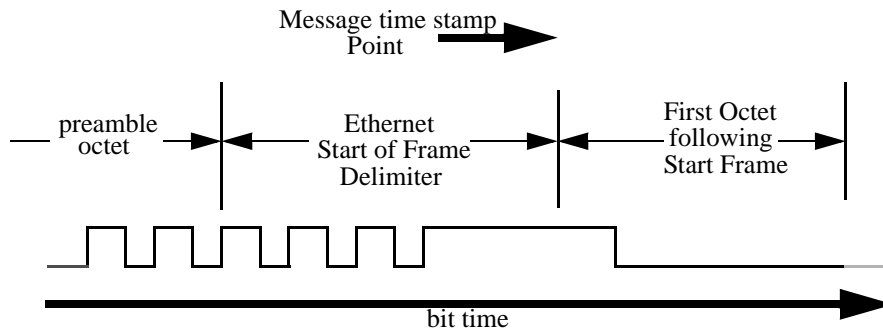


Figure 510. Message time stamp point with SFD detection

In addition, the TSU detects a Precise Time Protocol (PTP) frame of specific type and generates an indication to the CPU and to the MAC.

In the IEEE1588 V2 specification, there are ten types of PTP frames:

- Sync message
- Follow_Up message
- Delay_Req message
- Delay_Resp message
- Management message
- Pdelay_Req message
- Pdelay_Resp message
- Announce message
- Pdelay_Resp_Follow_Up message
- Signaling message

Those ten messages are divided into two groups:

- PTP event
- PTP general

The PTP event group consists of Sync, Delay_req, Pdelay_Req, and Pdealy_Resp messages. These messages should be time stamped and can cause an interrupt generation. The PTP general group consists of the rest of the frames, these frames transfer time stamps and determine general system parameters.

In order to detect a PTP frame see [Table 506](#) and [Table 508](#) for the parsing fields and values.

37.10.1 PTP Event Interrupts

In the case of out-of-band mode, the software can use the TSU interrupts as a trigger to read the time stamps registers. The software should clear the event register and read the time stamp before the next PTP frame detection. If a new PTP frame arrives and the relevant bit in the event register is still set, an overrun indication will be asserted.

37.11 IEEE1588 Real Time Clock (RTC)

In a distributed control system containing multiple clocks, individual clocks tend to drift apart due to instabilities inherent in source oscillators and environmental conditions such as temperature, air circulation, mechanical stresses, vibration, aging, etc. Hence, some kind of correction is necessary to synchronize individual clocks to maintain the notion of global time, which is accurate to some requisite clock resolution.

In order to achieve this goal, the IEEE1588 RTC supports the following:

- The nominal increment is chosen according to the nominal oscillator frequency
- The drift is compensated by slightly increasing or decreasing the increment

Figure 511 illustrates the block diagram of the IEEE1588 RTC.

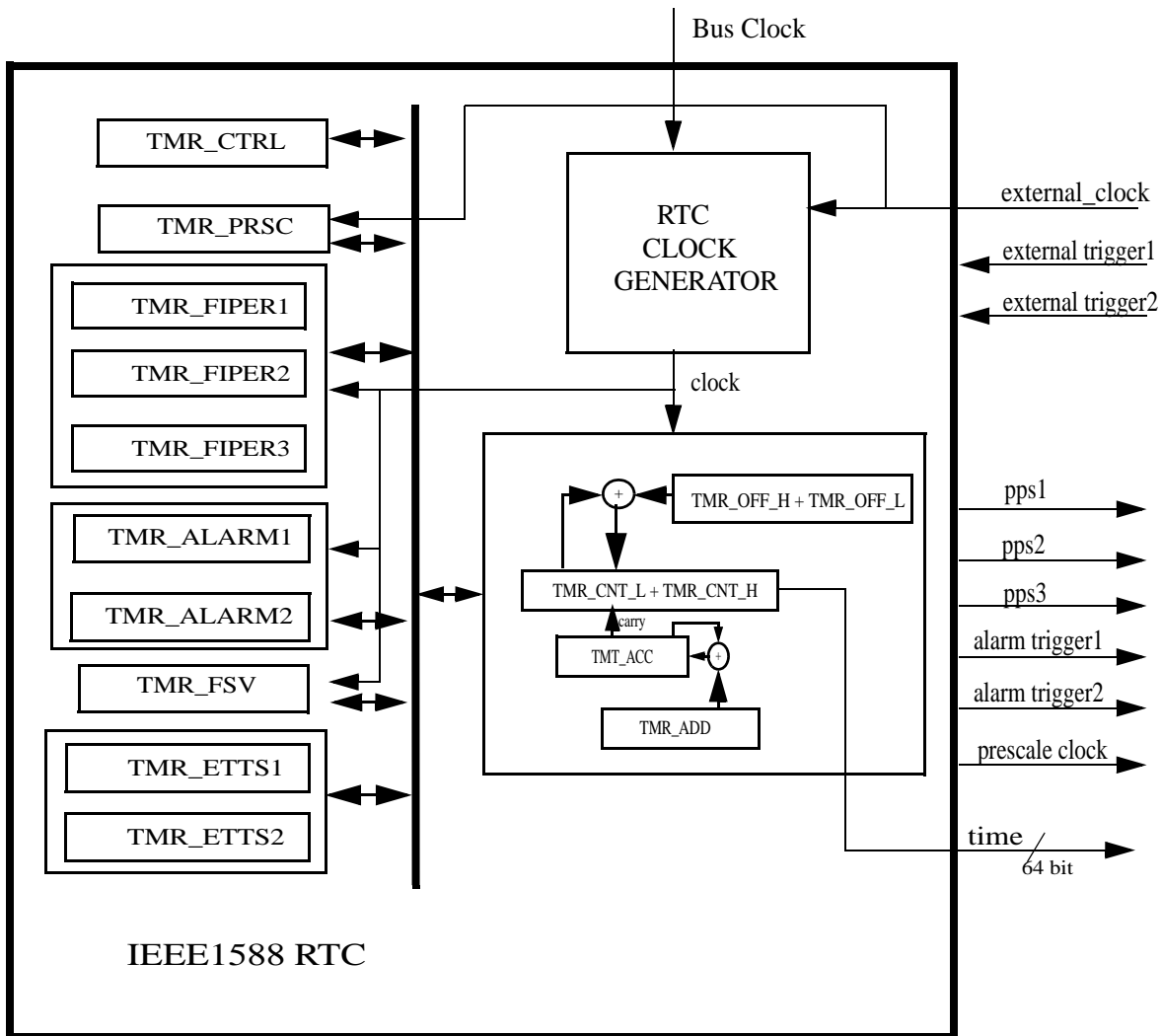


Figure 511. IEEE1588 RTC block diagram

All of the constituents of the frequency compensated clock operate at the frequency of the external clock source. The frequency compensation value contained in the addend register is added to the accumulator once every 1/external clock. The clock counter is incremented whenever the accumulator asserts an overflow pulse signal. Therefore, the nominal frequency at which the clock counter is incremented is the frequency that the software determined by updating the addend with a frequency compensation value.

The update of the frequency compensation value can be done every cycle of clock synchronization by Sync and delay messages.

Table 524 shows examples of possible IEEE1588 RTC parameters. The results have been calculated according to the this equation:

$$\text{frequency_compensation_value} = 2^{32}/\text{frequency_ratio}.$$

Table 524. Values Example

Frequency of the Clock	Frequency of the Counter	Frequency Ratio	Frequency Compensation Value
200MHz	150MHz	1.33	0xc07b301e
200MHz	190MHz	1.05	0xf3cf3cf3
200MHz	70MHz	2.86	0x5982af70

In addition to the frequency compensation, the RTC enables updating the offset of the timer counter. The offset value is derived from the time stamps transferred on PTP frames.

The software uses the offset register in order to add the correction value. The timer counter adds the value of the offset register if a write transaction to the relevant address is detected. The values in the offset register and in the addend register are represented in 2's complement.

The standard demands sub-microseconds resolution. The IEEE1588 RTC contains a 64-bit timer register and a 32-bit addend register. If the clock reaches 200MHz, by configuring the addend to 0xffffffff, the counter will tick every 5ns.

IEEE1588 RTC supports the following features:

- Two alarm output triggers, asserted if the timer value reaches the alarm (future time) register value
- Three pulse per second output signals, generated by configuring the FIPER registers. The signals are aligned in phase and frequency to the timer, and edge aligned to the IEEE1588 timer reference output clock (prescale clock). The PPSs signal width is according to the period of the selected timer clock or according to the width of the IEEE1588 timer reference output clock
- Two input triggers in order to capture time stamps in a dedicated register
- IEEE1588 timer reference output clock which is a divided output clock, by configuring the prescale register
- Input of a time stamp lock trigger to sample the IEEE1588 RTC according to an external trigger

37.11.1 RTC Clock Sources

The IEEE1588 RTC enables three sources of input clock: an external oscillator, PLL3, PLL4 , or the FEC_PHY clock.

All the clocks should meet the performances and requirements of the standard.

37.11.2 Prescale Output Clock and Pulse per Second Edge Alignment

In order to generate pulse per second signals that are edge aligned to the output prescale clock, as shown in Figure 512, the software configuration should be according to the following:

1. The ratio between the prescale clock divider value and the FIPER value should be divisible by the clock period, see Section 37.9.9, “Timer Fixed Interval Period Register (TMR_FIPERn)”.
2. Configure the TMR_FSV register to a value which is less by 3 clock periods of the required FIPER enabling time.
3. Note that each time when writing a new value to the TMR_FSV, the software should also write to the FIPERx register with the required PPS interval value.

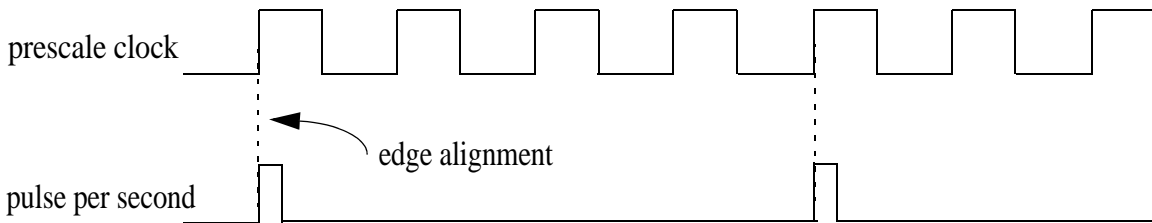


Figure 512. PPS and Prescale Clock Edge Alignment

37.12 PTP Frame Reception

The following describes the main actions performed for each operation mode when a PTP frame was received.

37.12.1 Out-of-Band Mode

In order to work in out-of-band mode the FEC should be configured as follows:

- FEC in Ethernet Mode
- Set IEEE1588 mode in UPSMR[PTPE]

The Rx TSU is monitoring the received Ethernet frame on the physical line. When SFD is detected or external time stamp lock trigger is received, the IEEE1588 Rx TSU locks the real time value of the IEEE1588 clock.

If the Rx TSU parse unit has detected a PTP frame, it will notify the Ethernet MAC that a PTP frame has been detected, and the Ethernet controller will set RxBD[RPTP]. In addition, the Rx TSU will set the corresponding bit in the event register and generate an interrupt to the CPU.

When the software detects that a PTP frame has arrived, either by getting an interrupt or by polling the RxBD[RPTP], it will read the dedicated time stamp register (TMR_UCx_RXTS_H/L) through the host interface.

In case the PTP frame is marked with CRC, Overrun, or Non-octet receive errors, the interrupt in the TMR_PEVENT will not be set and the software should drop this frame regardless of the indication in the RxBD[RPTP].

37.13 PTP Frame Transmission

The following describes the main actions performed for each operation mode when a PTP frame is transmitted.

[Timer PTP Event Register \(PTP_TMR_PEVENT\)/ Timer PTP Mask Register \(PTP_TMR_PEMASK\)](#) reports events recognized on the Ethernet channel and generates interrupts. After event recognition, the TSU sets the corresponding event bit. The event bits are cleared by writing 1. Writing a 0 does not affect bit values. All unmasked bits must be cleared before the CPU clears the internal interrupt request.

Interrupts generated by the event register can be masked in the mask register, which has the same bit format as the event register. If a mask bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked.

37.14 Cycle Delay from Time Stamp Location

The following table describes the delay on the Ethernet line, in clock cycles, for each of the interfaces between the Media Access Controller (MAC) devices and the physical layer devices (PHY) in Tx and Rx.

[Table 525](#) describes the delay on the Ethernet line.

Table 525. Delay on Ethernet Line

MII	
Tx	Rx
0	1

37.15 Initialization Sequence

37.15.1 TSU Mode Registers

1. Set the PTP_TSMR[OPMODE] bit to in-band or out-of-band mode.
2. Program the TMR_PEMASK register.
3. Write 0xFFFF_FFFF to TMR_PEVENT to clear the PTP Event register.

37.15.2 RTC Mode Registers

1. Initialize the TMR_ALARMn registers to 0xFFFF_FFFF
2. Initialize the TMR_FSV register to 0xFFFF_FFFF
3. Program the IEEE1588 timer clock period in TMR_CTRL[tlck_period]
4. Select the timer clock source in the TMR_CTRL[clkssel]
5. Program the TMR_TEMASK register
6. Write 0xFFFF_FFFF to TMR_TEVENT to clear the RTC Event register
7. Program the TMR_ADD according to the needed drift compensation needed value

37.15.3 Enable Sequence

1. Enable the RTC by setting TMR_CTRL[TE]
2. Enable the PTP by setting PTP_TSMR[EN]
3. Enable the FEC

Chapter 38

Nexus Debug Interface (NDI)

38.1 Overview

The NDI (Nexus Debug Interface) block provides real-time development support capabilities for the MPC5604E device in compliance with the IEEEISTO 5001-2003 standard. This development support is supplied for MCUs without requiring external address and data pins for internal visibility. The NDI block is an integration of several individual Nexus blocks that are selected to provide the development support interface for this device. The NDI block interfaces to the host processor and internal busses to provide development support as per the IEEE-ISTO 5001-2003 Class 2+ standard. The development support provided includes access to the MCUs internal memory map and access to the processors internal registers during run time.

38.1.1 Features

- Configured via the IEEE 1149.1
- All Nexus port pins operate at VDDIO (no dedicated power supply)
- Nexus 2+ features supported
- Static debug
- Watchpoint messaging
- Ownership trace messaging
- Program trace messaging
- Real time read/write of any internally memory mapped resources through JTAG pins
- Overrun control, which selects whether to stall before Nexus overruns or keep executing and allow overwrite of information
- Watchpoint triggering, watchpoint triggers program tracing
- Auxiliary Output Port
 - Four MDO (Message Data Out) pins in full port mode
 - MCKO (Message Clock Out) pin
 - Two MSEO (Message Start/End Out) pins
 - EVTO (Event Out) pin
- Auxiliary Input Port
 - EVTI (Event In) pin

38.2 Nexus Port Controller (NPC)

Figure 513 is a block diagram of the Nexus Port Controller (NPC) block.

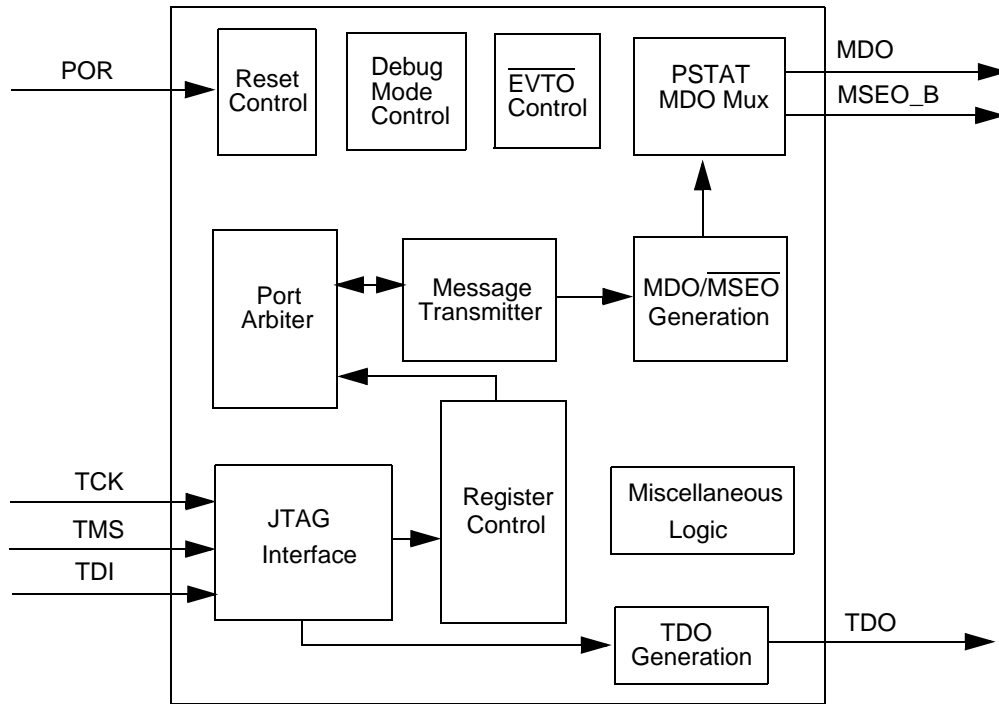


Figure 513. Nexus Port Controller Block Diagram

38.2.1 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Auxiliary Output Port
- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of $\overline{\text{EVTO}}$
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus blocks based on over-on reset status

38.2.2 Modes of Operation

The NPC block uses the power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

38.2.2.1 Reset

The NPC block is asynchronously placed in reset when power-on reset is asserted. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The auxiliary output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset)
- Registers default back to their reset values

38.2.2.2 Disabled-Port Mode

In disabled-port mode, auxiliary output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through EVTO.

38.2.2.3 Full-Port Mode

Full-port mode (FPM) is entered by asserting the MCKO_EN and FPM bits in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

38.3 External Signal Description

38.3.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in [Table 526](#).

Table 526. NPC Signal Properties

Name	Port	Function	Reset State	Pull ¹
EVTO_B	Auxiliary	Event Out pin	0b1	—
MDO	Auxiliary	Message Data Out pins	0 ²	—
MSEO	Auxiliary	Message Start/End Out pins	0b11	—
TCK	JTAG	Test Clock Input	—	Up
TDI	JTAG	Test Data Input	—	Up
TDO	JTAG	Test Data Output	High Z ³	—
TMS	JTAG	Test Mode Select Input	—	Up

¹ The pull is not implemented in this block. Pullup/pulldown devices are implemented in the pads.

² Following a power-on reset, MDO[0] remains asserted until power-on reset is exited and the system clock achieves lock.

- 3 TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

38.3.2 Detailed Signal Descriptions

This section describes each of the signals listed in [Table 526](#) in more detail. Note that the JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

38.3.2.1 EVTO_B - Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

38.3.2.2 MDO - Message Data Out

Message Data Out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

38.3.2.3 MSEO_B - Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

38.3.2.4 TCK - Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

NOTE

Due to limitation in the core, it is recommended to limit the frequency of TCK to be less than half of the System clock. Our system clock can be 8 MHz when sys clock divider is div-By-2 and sys clock is IRC16M. In all those scenarios, either the software needs to program the divider to be div-by-1 or the TCK is to be limited to 4MHz.

38.3.2.5 TDI - Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

38.3.2.6 TDO - Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

38.3.2.7 TMS - Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

38.4 Register Definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

[Table 527](#) shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers. Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 527. NPC Registers

Index	Register
0	Device ID Register (DID)
127	Port Configuration Register (PCR)

38.4.1 Register Descriptions

This section consists of NPC register descriptions.

38.4.1.1 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

38.4.1.2 Instruction Register

The NPC block uses a 4-bit instruction register as shown in [Table 514](#). The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state

results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

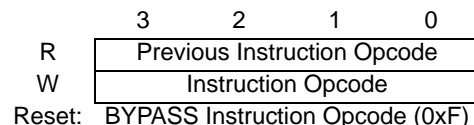


Figure 514. 4-Bit Instruction Register

38.4.1.3 Nexus Device ID Register (DID)

The device identification register, shown in [Figure 515](#), allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the auxiliary output port.

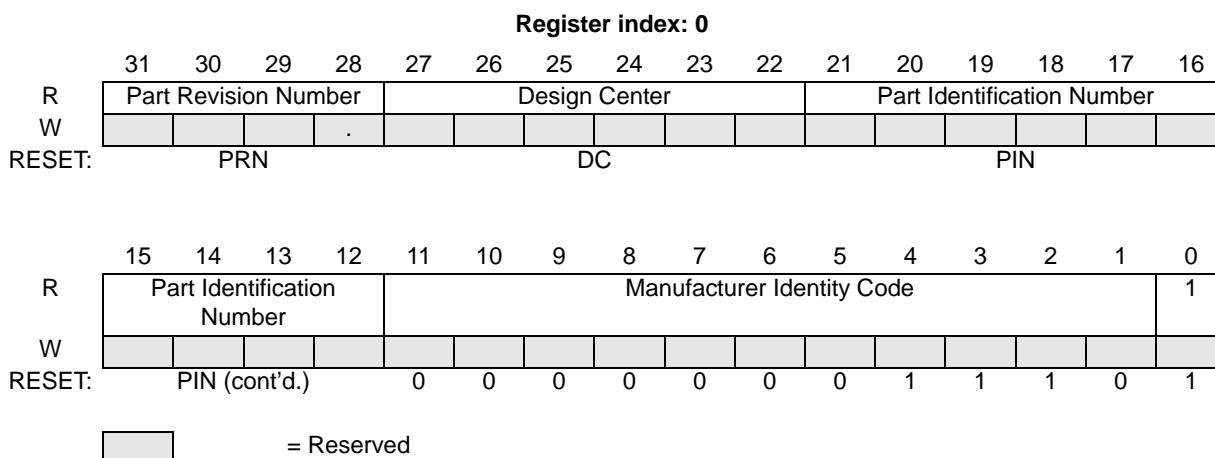


Figure 515. Nexus Device ID Register

Table 528. DID Field Descriptions

Bit	Name	Description
31:2 8	PRN	Part Revision Number These bits contain the revision number of the part
27:2 2	DC	Design Center These bits indicate the device design center For MPC5604E, this value is 011010.
21:1 2	PIN	Part Identification Number These bits contain the part number of the device For MPC5604E, this value is 1101_00000.

Table 528. DID Field Descriptions (continued)

Bit	Name	Description
11:1	MIC	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale Semiconductor, 0xE.
0	Bit [0]	IDCODE Register ID Bit [0] identifies this register as the device identification register and not the bypass register

38.4.1.4 Port Configuration Register (PCR)

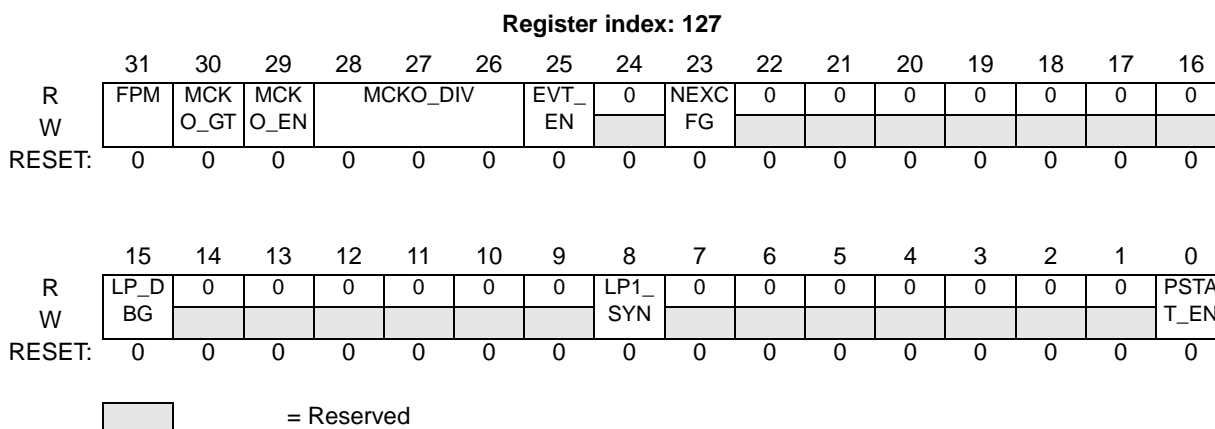


Figure 516. Port Configuration Register (PCR)

The PCR, shown in [Figure 516](#), is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

The PCR register may be rewritten by the debug tool subsequent to the enabling of the NPC for low power debug support. In this case, the debug tool may set and clear the LP_DBG and LPn_SYN bits, but must preserve the original state of the remaining bits in the register.

NOTE

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Table 529. PCR Field Descriptions

Bit	Name	Description
31	FPM	Full Port Mode The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 1 = All MDO pins are used to transmit messages

Table 529. PCR Field Descriptions (continued)

Bit	Name	Description
30	MCKO_GT	<p>MCKO Clock Gating Control</p> <p>This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted.</p> <p>1 = MCKO gating is enabled 0 = MCKO gating is disabled</p>
29	MCKO_EN	<p>MCKO Enable</p> <p>This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field.</p> <p>1 = MCKO clock is enabled 0 = MCKO clock is driven to zero</p>
28:2 6	MCKO_DIV	<p>MCKO Division Factor</p> <p>The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. Table 530 shows the meaning of MCKO_DIV Values. In this table, SYS_CLK represents the system clock frequency.</p>
25	EVT_EN	<p>EVTO/EVTI Enable</p> <p>This bit enables the EVTO/EVTI port functions.</p> <p>1 = EVTO/EVTI port enabled 0 = EVTO/EVTI port disabled</p>
23	NEXCFG	<p>Nexus Configuration Select</p> <p>Generic Nexus control bit. Function is SoC specific.</p> <p>1 = NEXCFG set 0 = NEXCFG cleared</p>
22:1 6	-	Reserved
15	LP_DBG_EN	<p>Low Power Debug Enable</p> <p>This bit enables debug functionality on exit from low power modes on supported devices.</p> <p>1 = Low power debug enabled 0 = Low power debug disabled</p>
14:1 0	-	Reserved
9:8	LPn_SYN	<p>Low Power Mode n Synchronization</p> <p>These bits are used to synchronize the entry into low power modes between the device and debug tool. Supported devices set these bits before a pending entry into low power mode. After reading the bit as set, the debug tool then clears the bit to acknowledge to the device that it may enter the low power mode.</p> <p>1 = Low power mode entry pending 0 = Low power mode entry acknowledged</p>
7:1	-	Reserved

Table 529. PCR Field Descriptions (continued)

Bit	Name	Description
0	PSTAT_EN	<p>Processor Status Mode Enable¹</p> <p>This bit enables processor status (PSTAT) mode. In PSTAT mode, all auxiliary output port MDO pins are used to transmit processor status information, and Nexus messaging is unavailable.</p> <p>1 = PSTAT mode enabled 0 = PSTAT mode disabled</p>

¹ PSTAT Mode is intended for factory processor debug only. The PSTAT_EN bit should be written to disable PSTAT mode if Nexus messaging is desired. No Nexus messages are transmitted under any circumstances when PSTAT mode is enabled.

Table 530. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO Frequency
0	SYS_CLK ¹
1	SYS_CLK/2
2	SYS_CLK/3
3	SYS_CLK/4
4	Reserved
5	Reserved
6	Reserved
7	SYS_CLK/8

¹ The SYS_CLK setting for MCKO frequency should only be used if this setting does not violate the maximum operating frequency of the auxiliary port pins.

38.5 Functional Description

38.5.1 NPC_HNDSHK module

This module enables debug entry/exit across low power modes(Stop, Halt, standby).The NPC_HNDSHK supports:

- Setting and clearing of the NPC PCR sync bit on low-power mode entry and exit
- Putting the core into debug mode on low-power mode exit
- Generating a falling edge on the JTAG TDO pad on low-power mode exit

On HALT0, STOP0 mode entry, the MC_ME asserts the lp_mode_entry_req input after the clock disable process has completed and before the processor enters its halted or stopped state. The mode transition will then not proceed until the lp_mode_entry_ack output has been asserted. The notification to the debugger of a low-power mode entry consists of setting the low-power mode handshake bit in the port control register (read by the debugger) via the lp_sync_in output. The debugger acknowledges that the transition into a low-power mode may proceed by clearing the low-power mode handshake bit in the port control register (written by the debugger), which results in the deassertion of the lp_sync_out input.

In anticipation of the low-power mode exit notification, the TDO pad is driven to `1'.On HALT0 or STOP0 mode exit, the MC_ME asserts the lp_mode_exit_req input after ensuring that the regulator and memories

are in normal mode and before the processor exits its halted or stopped state. The mode transition will then not proceed until the lp_mode_exit_ack output has been asserted. The MC_RGM asserts the exit_from_standby input when executing a reset sequence due to a STANDBY0 exit. The reset sequence will then not complete until the lp_mode_exit_ack output has been asserted. The notification to the debugger of a low-power mode exit consists of driving the TDO pad to `0'. The debugger acknowledges that the transition from a low-power mode can continue by setting the low-power mode sync bit in the port control register (written by debugger), which results in the assertion of the lp_sync_out input.

NOTE

The debugger clock multiplexer may not guarantee glitch free switching. Therefore, TCK should be disabled from when the debugger clears the sync bit in ENTRY_CLR until the debugger senses the falling edge of TDO in TDO_SET.

When debugging through stop/halt mode with handshaking enabled, reset should not be asserted for exiting the STOP/HALT mode.

38.5.2 NPC Reset Configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

Table 531 describes the NPC reset configuration options.

Table 531. NPC Reset Configuration Options

MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
X	X	Reset
0	X	Disabled
1	1	Full-Port Mode

38.5.3 Auxiliary Output Port

The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

38.5.3.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSEO}}$ functions. The $\overline{\text{MSEO}}$ pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and $\overline{\text{MSEO}}$ are sampled on the rising edge of MCKO.

Figure 517 illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

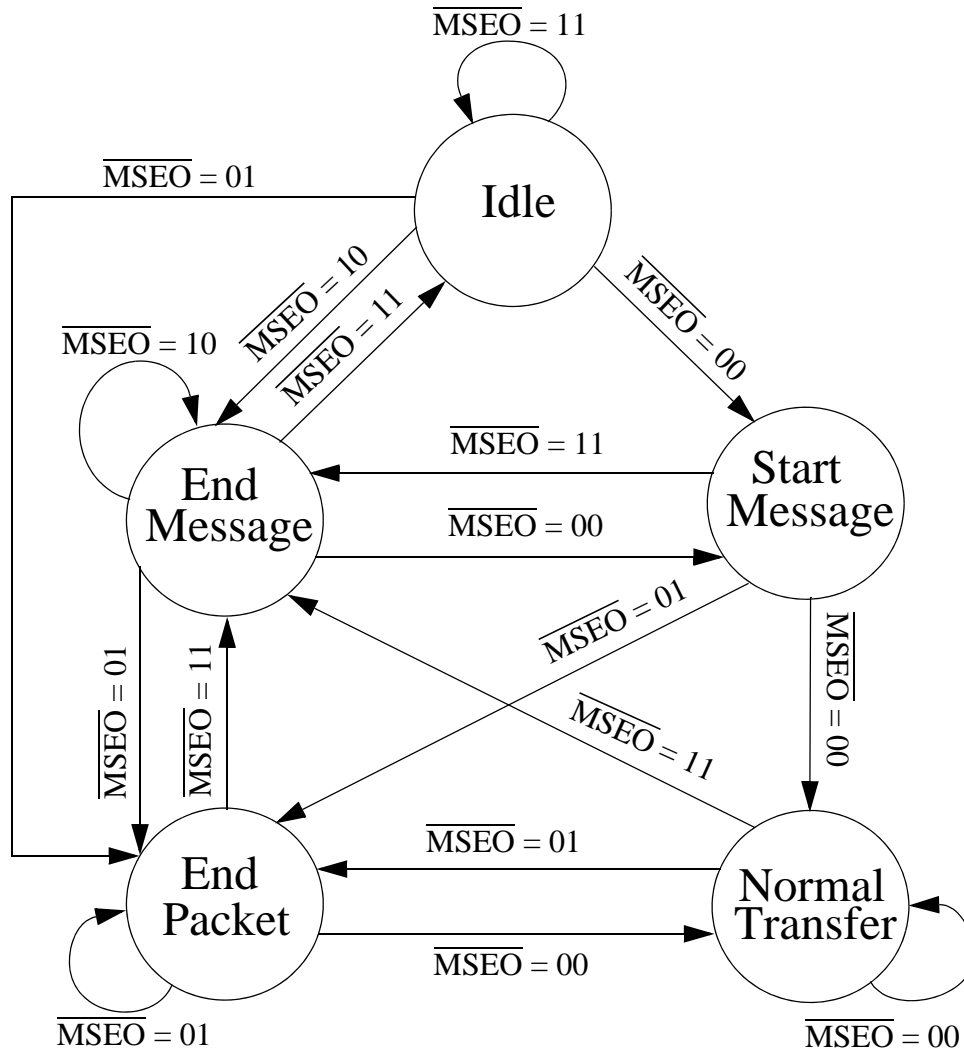


Figure 517. MSEO Transfers (for 2-bit MSEO)

38.5.3.2 Output Messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register and the port replacement output message on the MDO pins. The device ID message can also be sent out serially through TDO.

Table 532 describes the device ID and port replacement output messages that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Table 532. NPC Output Messages

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents

Figure 518 shows the various message formats that the pin interface formatter has to encounter. Note that for variable-length fields, the transmitted size of the field is determined from the range of the least significant bit to the most significant non-zero-valued bit (i.e. most significant zero-valued bits are not transmitted).

Message	TCODE	Field #1	Field #2	Field #3	Field #4	Field #5	Min. Size ¹ (bits)	Max Size ² (bits)
Device ID Message	1	Fixed = 32	NA	NA	NA	NA	38	38

NOTES:

1. Minimum information size. The actual number of bits transmitted depends on the number of MDO pins
2. Maximum information size. The actual number of bits transmitted depends on the number of MDO pins

Figure 518. Message Field Sizes

The double edges in Figure 518 indicate the starts and ends of messages. Fields without shaded areas between them are grouped into super-fields and can be transmitted together without end-of-packet indications between them.

38.5.3.3 Rules of Message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. Figure 519 shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

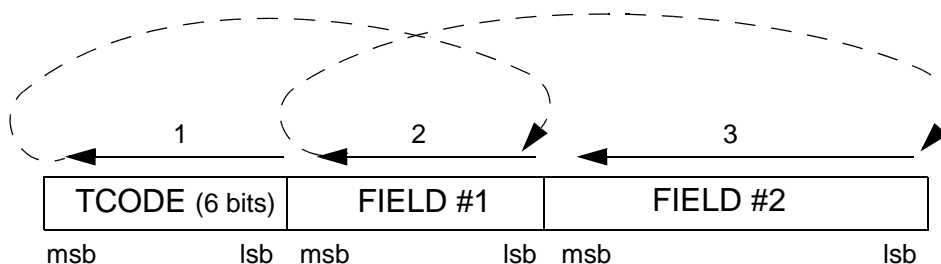


Figure 519. Transmission Sequence of Messages

38.5.4 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 TAP for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller.

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in [Figure 521](#). The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in [Figure 522](#).

The instructions implemented by the NPC TAP controller are listed in [Table 533](#). The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4-bits.

Table 533. Implemented Instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in [Figure 520](#). This applies for the instruction register and all Nexus tool-mapped registers.

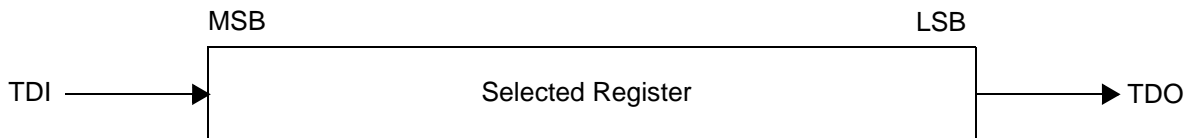
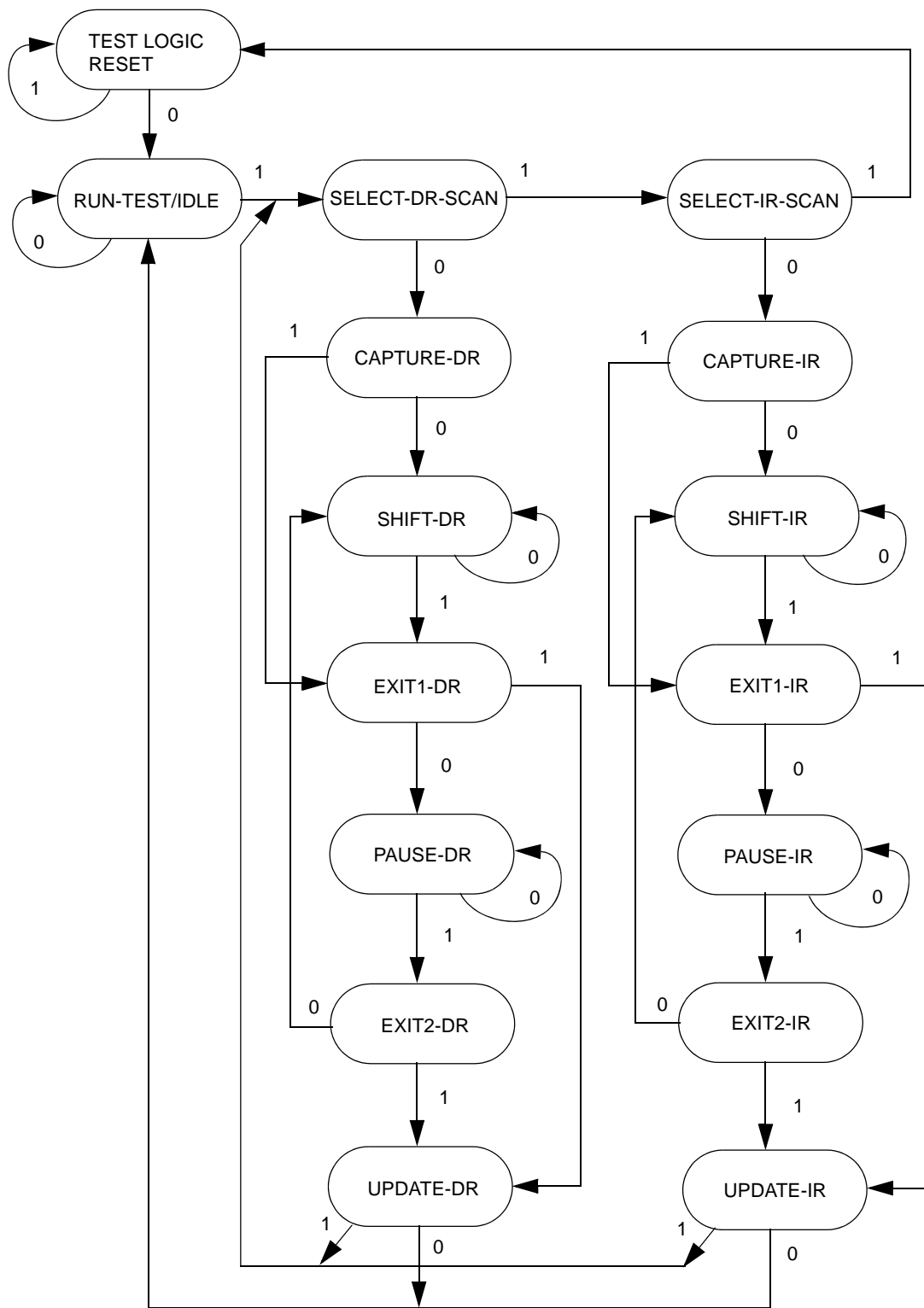


Figure 520. Shifting Data Into Register

38.5.4.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal resets the NPC TAP controller. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 521. IEEE 1149.1-2001 TAP Controller State Machine

38.5.4.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the auxiliary output port or shifted out serially by accessing the Nexus Device ID register through the TAP. Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled. Transmission of the device identification message serially via TDO is achieved by performing a read of the register contents as described in Section 38.5.4.4, “Selecting a Nexus Client Register”.

38.5.4.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 522, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. Table 534 illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

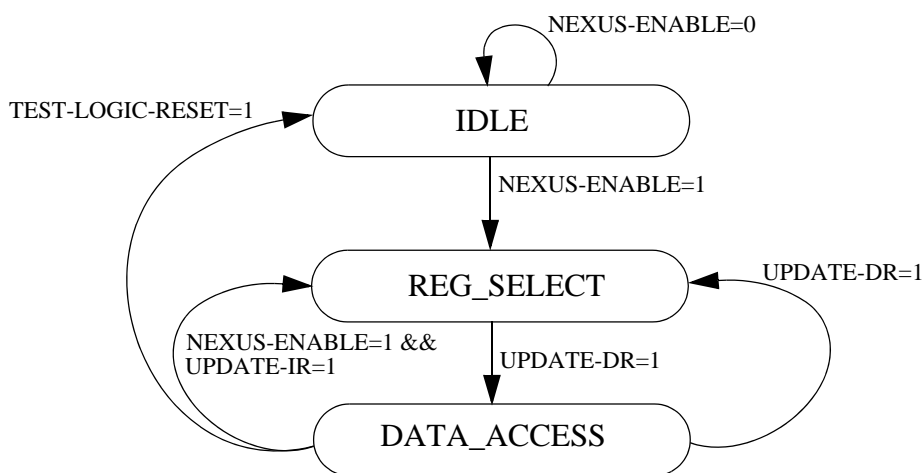


Figure 522. NEXUS Controller State Machine

Table 534. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

38.5.4.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in Figure 523. The read/write control bit is set to 1 for writes and 0 for reads.

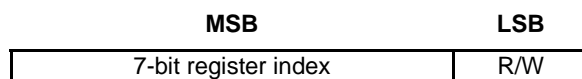


Figure 523. IEEE 1149.1 Controller Command Input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

Table 535 illustrates a sequence which writes a 32-bit value to a register

Table 535. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

38.5.5 Nexus JTAG Port Sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

38.5.6 MCKO and ipg_sync_mcko

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

38.5.7 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

38.5.8 Nexus Reset Control

If an internal reset is asserted, all Nexus modules should be held in reset.

38.6 Initialization/Application Information

38.6.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.

2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2003 standard for more detail.

Chapter 39

Register Protection (REG_PROT)

39.1 Introduction

39.1.1 Overview

The REG_PROT module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the AIPS. This is shown in [Figure 524](#).

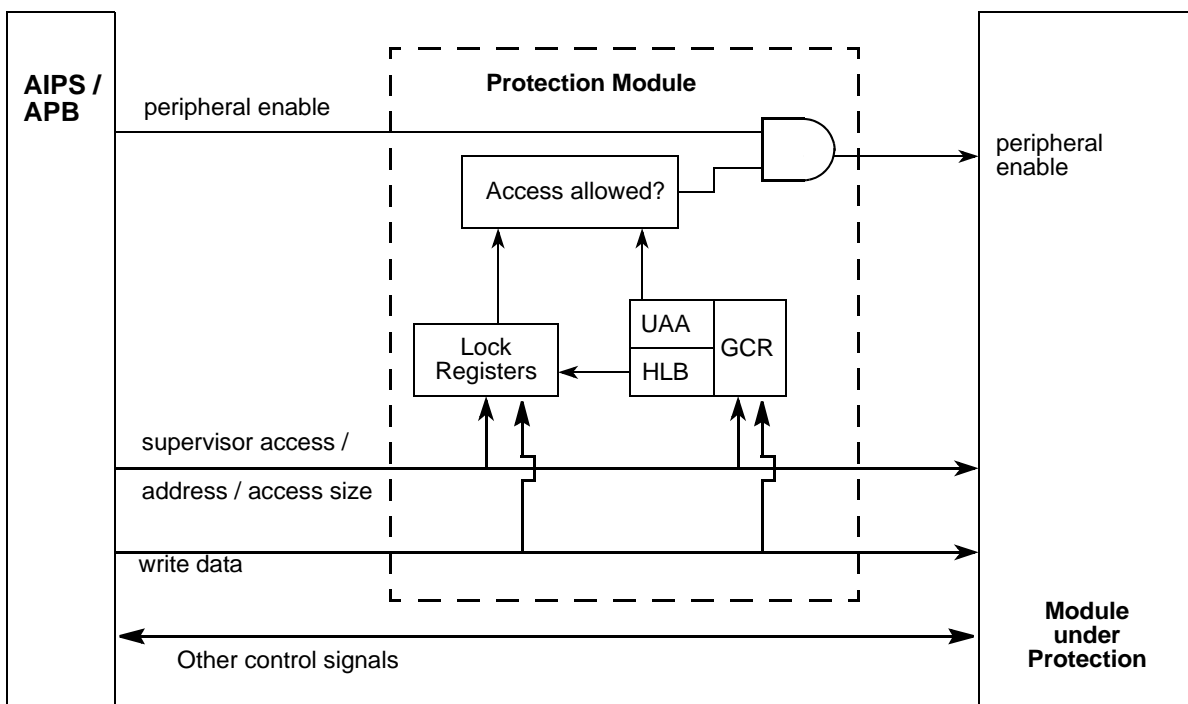


Figure 524. REG_PROT Block Diagram

39.1.2 Features

The Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only

- Lock registers for first 6KB of memory-mapped address space
- Address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

39.1.3 Modes of Operation

The Register Protection module is operable when the module under protection is operable. For further details about the availability please see the Block Guide of the according module.

39.2 External Signal Description

There are no external signals.

39.3 Memory Map and Register Definition

This section provides a detailed description of the memory map of a module using the REG_PROT. The original 16K module memory space is divided into 5 areas as shown in [Figure 525](#).

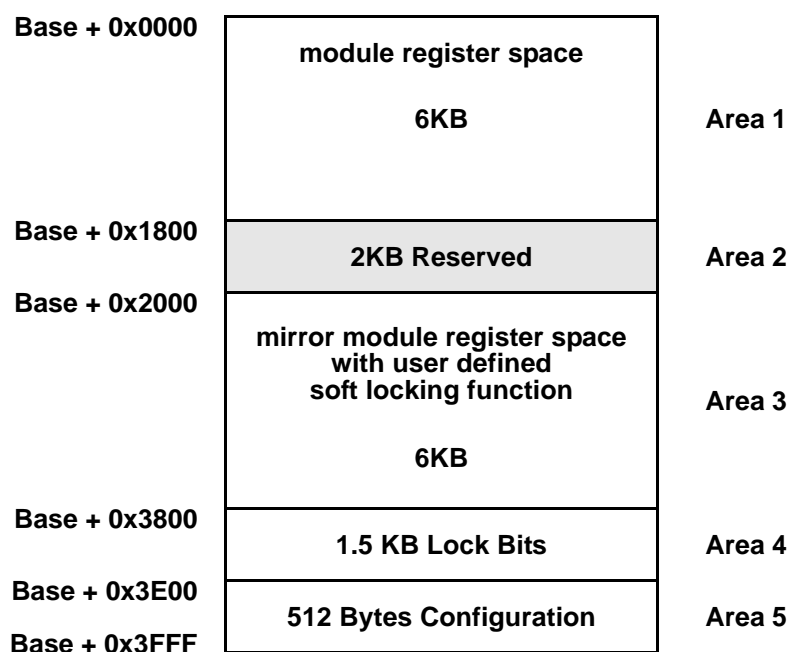


Figure 525. REG_PROT Memory Diagram

Area 1 is 6KB large and holds the normal functional module registers and is transparent for all read/write operations.

Area 2 is 2KB starting at address 0x1800 is a reserved area, which shall not be accessed.

Area 3 is 6KB large, starting at address 0x2000 and is a mirror of area 1. A read/write access to these 0x2000+X addresses will read/write the register at address X. As a side effect, a write access to address

0x2000+X will set the optional Soft Lock Bits for this address X in the same cycle as the register at address X is written. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at address Y, accesses to address 0x2000+Y will be identical to accesses at address Y. Only for registers implemented in area 1 and defined as protectable Soft Lock Bits will be available in area 4.

Area 4 is 1.5KB large and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.

Area 5 is 512 Bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-Bit registers in Areas 4 and 5 will result in a transfer error.

39.3.1 Memory Map

Figure 536 gives an overview on the REG_PROT registers implemented.

Table 536. PROTECTED MODULE Memory Map

Address Offset	Use	Section/Page
0x0000	Module Register 0 (MR0)	39.3.2.1/934
0x0001	Module Register 1 (MR1)	39.3.2.1/934
0x0002	Module Register 2 (MR2)	39.3.2.1/934
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	39.3.2.1/934
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	39.3.2.2/934
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	39.3.2.2/934
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	39.3.2.2/934
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	39.3.2.3/934
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	39.3.2.3/934
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	39.3.2.3/934
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	39.3.2.4/935

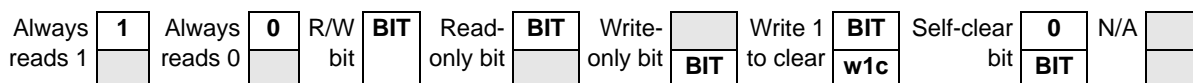
NOTE

Reserved registers in area #2 will be handled according to the protected IP (module under protection).

39.3.2 Register Descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

Figure 526. Key to Register Fields



39.3.2.1 Module Registers (MR0-6143)

This is the lower 6K module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

39.3.2.2 Module Register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBRn.SLBm, according to the mapping described in [Table 537](#).

39.3.2.3 Soft Lock Bit Register (SLBR0-1535)

These registers hold the Soft Lock Bits for the protected registers in memory area #1.

Address 0x3800-0x3DFF

Access: Read always
Supervisor write

	0	1	2	3	4	5	6	7
R	0	0	0	0	SLB0	SLB1	SLB2	SLB3
W	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

Figure 527. Soft Lock Bit Register (SLBRn)

Table 537. SLBR_n Field Descriptions

Field	Description
WE0 WE1 WE2 WE3	Write Enable Bits for Soft Lock Bits (SLB): WE0 enables writing to SLB0 WE1 enables writing to SLB1 WE2 enables writing to SLB2 WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
SLB0 SLB1 SLB2 SLB3	Soft Lock Bits for one MR _n register: SLB0 can block accesses to MR[<i>n</i> * 4 + 0] SLB1 can block accesses to MR[<i>n</i> * 4 + 1] SLB2 can block accesses to MR[<i>n</i> * 4 + 2] SLB3 can block accesses to MR[<i>n</i> * 4 + 3] 1 Associated MR _n byte is locked against write accesses 0 Associated MR _n byte is unprotected and writeable

Table 538. gives some examples how SLBR_n.SLB and MR_n go together:

Table 538. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

39.3.2.4 Global Configuration Register (GCR)

This register is used to make global configurations related with the REG_PROT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	H	0	0	0	0	0	0	0	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	L								A																								
B									A																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 528. Global Configuration Register (GCR)

Table 539. GCR Field Descriptions

Field	Description
HLB	<p>Hard Lock Bit. This register can not be cleared once it is set by software. It can only be cleared by a system reset.</p> <p>1 All SLB bits are write protected and can not be modified 0 All SLB bits are accessible and can be modified.</p>
UAA	<p>User Access Allowed.</p> <p>1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.</p>

NOTE

The GCR.UAA bit has no effect on the allowed access modes for the registers in the REG_PROT module.

39.4 Functional Description

39.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)
- unprotected (address == multiples of 1)

Which addresses are protected and the protection size depend on the SoC and/or module. Therefore this Block Guide can just give examples for various protection configurations.

For all addresses that are protected there are SLBR_n.SLB_m bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBR_n.SLB_m bit is always 0b0 no matter what software is writing to.

39.4.2 Change Lock Settings

To change the setting whether an address is locked or unlocked the corresponding $SLBR_n.SLB_m$ bit needs to be changed. This can be done using the following methods:

- Modify the $SLBR_n.SLB_m$ directly by writing to area #4
- Set the $SLBR_n.SLB_m$ bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

39.4.2.1 Change Lock Settings Directly Via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each $SLBR_n.SLB_m$ bit has a mask bit $SLBR_n.WE_m$ which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 529. shows two modification examples. In the left example there is a write access to the $SLBR_n$ register specifying a mask value which allows modification of all $SLBR_n.SLB_m$ bits. The example on the right specifies a mask which only allows modification of the bits $SLBR_n.SLB[3:1]$.

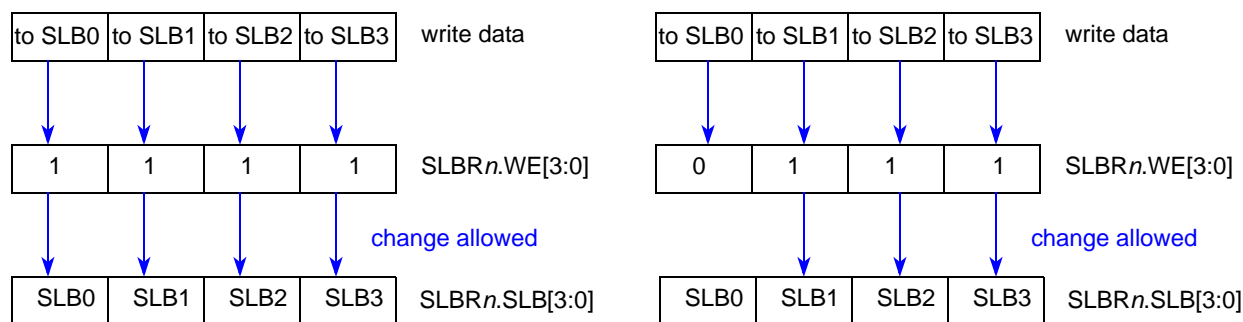


Figure 529. Change Lock Settings Directly Via Area #4

Figure 529. showed four registers that can be protected 8-bit wise. In **Figure 530.** registers with 16-bit protection and in **Figure 531.** registers with 32-bit protection are shown:

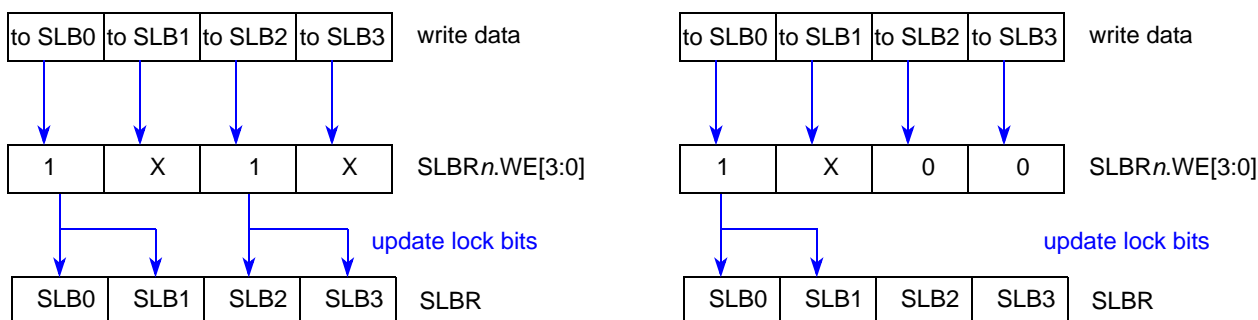


Figure 530. Change Lock Settings for 16-bit Protected Addresses

On the right side of **Figure 530**, it is shown that the data written to $SLBRn.SLB[0]$ is automatically written to $SLBRn.SLB[1]$ also. This is done as the address reflected by $SLBRn.SLB[0]$ is protected 16-bit wise. Note that in this case the write enable $SLBRn.WE[0]$ must be set while $SLBRn.WE[1]$ does not matter. As the enable bits $SLBRn.WE[3:2]$ are cleared the lock bits $SLBRn.SLB[3:2]$ remain unchanged.

In the example on the left side of **Figure 530**, the data written to $SLBRn.SLB[0]$ is mirrored to $SLBRn.SLB[1]$ and the data written to $SLBRn.SLB[2]$ is mirrored to $SLBRn.SLB[3]$ as for both registers the write enables are set.

In **Figure 531**, a 32-bit wise protected register is shown. When $SLBRn.WE[0]$ is set the data written to $SLBRn.SLB[0]$ is automatically written to $SLBRn.SLB[3:1]$ also. Otherwise $SLBRn.SLB[3:0]$ remains unchanged.

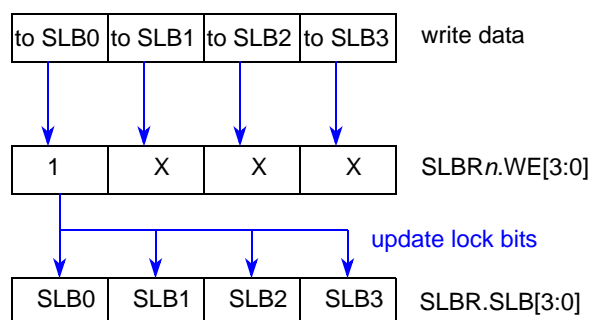


Figure 531. Change Lock Settings for 32-bit Protected Addresses

In **Figure 532**, an example is shown which has a mixed protection size configuration:

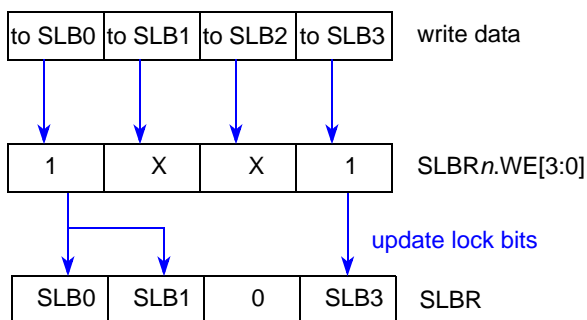


Figure 532. Change Lock Settings for Mixed Protection

The data written to $SLBRn.SLB[0]$ is mirrored to $SLBRn.SLB[1]$ as the corresponding register is 16-bit protected. The data written to $SLBRn.SLB[2]$ is blocked as the corresponding register is unprotected. The data written to $SLBRn.SLB[3]$ is written to $SLBRn.SLB[3]$.

39.4.2.2 Enable Locking Via Mirror Module Space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. **Figure 533** shows one example:

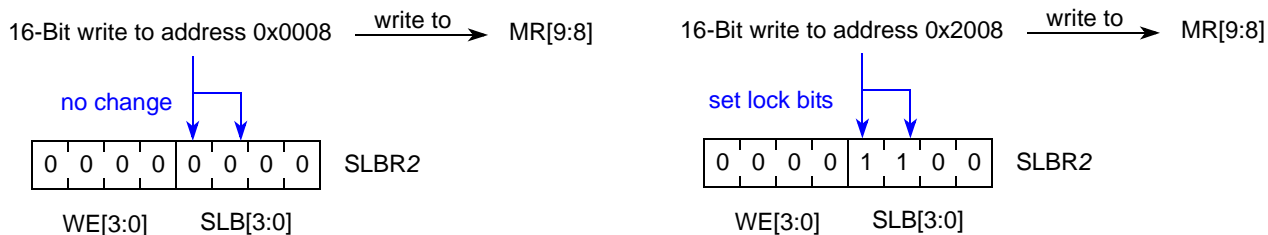


Figure 533. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of [Figure 530](#)).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of [Figure 530](#)).

[Figure 534](#) shows an example where some addresses are protected and some are not:

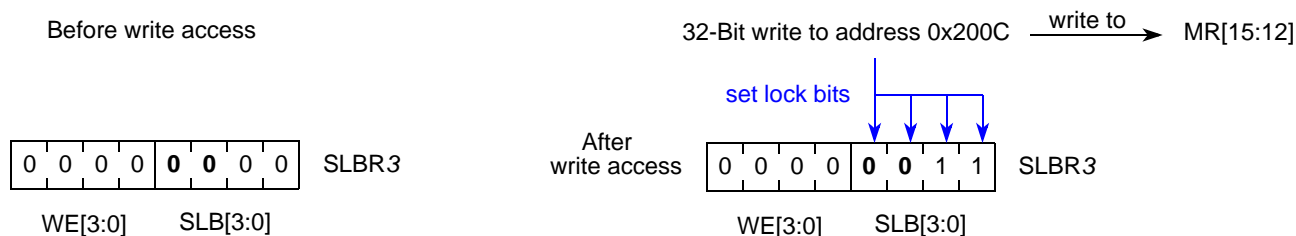


Figure 534. Enable Locking for Protected and Unprotected Addresses

In the example in [Figure 534](#), addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

NOTE

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

39.4.2.3 Write Protection for Locking Bits

Changing the locking bits through any of the procedures mentioned in [Section 39.4.2.1, “Change Lock Settings Directly Via Area #4”](#) and [Section 39.4.2.2, “Enable Locking Via Mirror Module Space \(Area #3\)”](#) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there was a system reset.

39.4.3 Access Errors

The protection module generates transfer errors under several circumstances. For the area definition refer to [Figure 525](#).

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set.

39.5 Initialization/Application Information

39.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See [Section 39.3.2, “Register Descriptions”](#)). In summary, after reset, locking for all MR n registers is disabled. The registers can be accessed in Supervisor Mode only.

39.5.2 Writing C code using the register protection scheme

There is a set of macros provided as part of the device specific header file, which defines the memory map, the peripheral registers, and the fields of these registers. These macros are intended to make working with the register protection scheme easier for the developers of device driver software and/or other application code. This section describes these macros and how to use them.

A first macro is made available to perform a write to the mirrored module register space (Area 3). As described in this document, this results in concurrently setting the corresponding soft lock bit, while writing to the module register. There are three flavors of this macro, to account for the different size of the related module register `<thereg>` (the value of `<size>` is either 8, 16, or 32 bit):

```
WRITE_WITH_LOCK<size>(<thereg>, <newvalue>)
```

This macro writes the value `<newvalue>` into the register `<thereg>` assuming a register size of `<size>`. The parameter `<thereg>` must be the name of a register using the notation in the device specific header file.

A second set of macros is made available to work with the soft lock bits provided by the register protection scheme. The value of these bits associated with a particular module register `<thereg>` can be retrieved, set, and cleared. The macros provided for this purpose are:

```
GET_SOFTLOCK (<thereg>, <dest>)
```



```
SET_SOFTLOCK<size>(<thereg>)  
CLR_SOFTLOCK<size>(<thereg>)
```

The macro GET_SOFTLOCK retrieves the value of the soft lock bit register associated with the given register <thereg> and stores it in the variable <dest>; which is always an eight bit value. The other two macros (SET_SOFTLOCK, CLR_SOFTLOCK) set or clear the softlock bits associated with the given register <thereg>; assuming this register has a size of 8, 16, or 32 bit as indicated by the macro name. For all three macros, the parameter <thereg> must be the name of a register using the notation in the device specific header file.

Three more macros are made available to modify bits in the Global Configuration Register (GCR) of the protection gasket for the corresponding block. In contrast to the earlier ones, these macros receive the base address of the corresponding block (usually named <block_name>_BASEADDRESS) as a parameter.

```
SET_HARDLOCK(base)
```

Sets the Hard Lock Bit HLB in the GCR register associated with the module identified by the given base address <base>.

```
USER_ACCESS_FORBIDDEN(base)
```

Clears the User Access Allowed Bit UAA in the GCR register associated with the module identified by the given base address <base>. When cleared, this bit denies any write access to the protected module in user mode and generates a transfer error in case of an attempt to write a protected register.

```
USER_ACCESS_ALLOWED(base)
```

Sets the User Access Allowed Bit UAA in the GCR register associated with the module identified by the given base address <base>. When set, this bit permits write accesses to the protected module in user mode when the corresponding register are not additionally protected by other means.

Finally, two more macros are provided to permit direct access to the registers in the register protection gasket using the same semantic as other register definitions in the device specific header file:

```
LOCK_SLB(thereg)
```

provides the content of the soft lock bit register associated with the register <thereg>; the parameter <thereg> must be the name of the corresponding module register using the notation in the device specific header file.

```
LOCK_GCR(base)
```

provides the content of the Global Configuration Register GCR for the block identified by its base address <base>; the parameter <base> must be the base address of the corresponding block.

39.6 Registers under protection

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
Code Flash						
Code Flash	MCR	32	C3F88000	000	C3F88000	32-bit
Code Flash	BIU0	32	C3F88000	01C	C3F8801C	32-bit
Code Flash	BIU1	32	C3F88000	020	C3F88020	32-bit
Code Flash	BIU2	32	C3F88000	024	C3F88024	32-bit
Data Flash						
Data Flash	MCR	32	C3F8C000	000	C3F8C000	32-bit
SIU lite						
SIUL	IRER	32	C3F90000	018	C3F90018	32-bit
SIUL	IREER	32	C3F90000	028	C3F90028	32-bit
SIUL	IFEER	32	C3F90000	02C	C3F9002C	32-bit
SIUL	IFER	32	C3F90000	030	C3F90030	32-bit
SIUL	PCR0	16	C3F90000	040	C3F90040	16-bit
SIUL	PCR1	16	C3F90000	042	C3F90042	16-bit
SIUL	PCR2	16	C3F90000	044	C3F90044	16-bit
SIUL	PCR3	16	C3F90000	046	C3F90046	16-bit
SIUL	PCR4	16	C3F90000	048	C3F90048	16-bit
SIUL	PCR5	16	C3F90000	04A	C3F9004A	16-bit
SIUL	PCR6	16	C3F90000	04C	C3F9004C	16-bit
SIUL	PCR7	16	C3F90000	04E	C3F9004E	16-bit
SIUL	PCR8	16	C3F90000	050	C3F90050	16-bit
SIUL	PCR9	16	C3F90000	052	C3F90052	16-bit
SIUL	PCR10	16	C3F90000	054	C3F90054	16-bit
SIUL	PCR11	16	C3F90000	056	C3F90056	16-bit
SIUL	PCR12	16	C3F90000	058	C3F90058	16-bit
SIUL	PCR13	16	C3F90000	05A	C3F9005A	16-bit
SIUL	PCR14	16	C3F90000	05C	C3F9005C	16-bit
SIUL	PCR15	16	C3F90000	05E	C3F9005E	16-bit
SIUL	PCR16	16	C3F90000	060	C3F90060	16-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
SIUL	PCR17	16	C3F90000	062	C3F90062	16-bit
SIUL	PCR18	16	C3F90000	064	C3F90064	16-bit
SIUL	PCR19	16	C3F90000	066	C3F90066	16-bit
SIUL	PCR20	16	C3F90000	068	C3F90068	16-bit
SIUL	PCR21	16	C3F90000	06A	C3F9006A	16-bit
SIUL	PCR22	16	C3F90000	06C	C3F9006C	16-bit
SIUL	PCR23	16	C3F90000	06E	C3F9006E	16-bit
SIUL	PCR24	16	C3F90000	070	C3F90070	16-bit
SIUL	PCR25	16	C3F90000	072	C3F90072	16-bit
SIUL	PCR26	16	C3F90000	074	C3F90074	16-bit
SIUL	PCR27	16	C3F90000	076	C3F90076	16-bit
SIUL	PCR28	16	C3F90000	078	C3F90078	16-bit
SIUL	PCR29	16	C3F90000	07A	C3F9007A	16-bit
SIUL	PCR30	16	C3F90000	07C	C3F9007C	16-bit
SIUL	PCR31	16	C3F90000	07E	C3F9007E	16-bit
SIUL	PCR32	16	C3F90000	080	C3F90080	16-bit
SIUL	PCR33	16	C3F90000	082	C3F90082	16-bit
SIUL	PCR34	16	C3F90000	084	C3F90084	16-bit
SIUL	PCR35	16	C3F90000	086	C3F90086	16-bit
SIUL	PCR36	16	C3F90000	088	C3F90088	16-bit
SIUL	PCR37	16	C3F90000	08A	C3F9008A	16-bit
SIUL	PCR38	16	C3F90000	08C	C3F9008C	16-bit
SIUL	PCR39	16	C3F90000	08E	C3F9008E	16-bit
SIUL	PCR40	16	C3F90000	090	C3F90090	16-bit
SIUL	PCR41	16	C3F90000	092	C3F90092	16-bit
SIUL	PCR42	16	C3F90000	094	C3F90094	16-bit
SIUL	PCR43	16	C3F90000	096	C3F90096	16-bit
SIUL	PCR44	16	C3F90000	098	C3F90098	16-bit
SIUL	PCR45	16	C3F90000	09A	C3F9009A	16-bit
SIUL	PCR46	16	C3F90000	09C	C3F9009C	16-bit
SIUL	PCR47	16	C3F90000	09E	C3F9009E	16-bit
SIUL	PCR48	16	C3F90000	0A0	C3F900A0	16-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
SIUL	PCR49	16	C3F90000	0A2	C3F900A2	16-bit
SIUL	PCR50	16	C3F90000	0A4	C3F900A4	16-bit
SIUL	PCR51	16	C3F90000	0A6	C3F900A6	16-bit
SIUL	PCR52	16	C3F90000	0A8	C3F900A8	16-bit
SIUL	PCR53	16	C3F90000	0AA	C3F900AA	16-bit
SIUL	PCR54	16	C3F90000	0AC	C3F900AC	16-bit
SIUL	PCR55	16	C3F90000	0AE	C3F900AE	16-bit
SIUL	PCR56	16	C3F90000	0B0	C3F900B0	16-bit
SIUL	PCR57	16	C3F90000	0B2	C3F900B2	16-bit
SIUL	PCR58	16	C3F90000	0B4	C3F900B4	16-bit
SIUL	PCR59	16	C3F90000	0B6	C3F900B6	16-bit
SIUL	PCR60	16	C3F90000	0B8	C3F900B8	16-bit
SIUL	PCR61	16	C3F90000	0BA	C3F900BA	16-bit
SIUL	PCR62	16	C3F90000	0BC	C3F900BC	16-bit
SIUL	PCR63	16	C3F90000	0BE	C3F900BE	16-bit
SIUL	PCR64	16	C3F90000	0C0	C3F900C0	16-bit
SIUL	PCR65	16	C3F90000	0C2	C3F900C2	16-bit
SIUL	PCR66	16	C3F90000	0C4	C3F900C4	16-bit
SIUL	PCR67	16	C3F90000	0C6	C3F900C6	16-bit
SIUL	PCR68	16	C3F90000	0C8	C3F900C8	16-bit
SIUL	PCR69	16	C3F90000	0CA	C3F900CA	16-bit
SIUL	PCR70	16	C3F90000	0CC	C3F900CC	16-bit
SIUL	PSMI0_3	32	C3F90000	500	C3F90500	32-bit
SIUL	PSMI4_7	32	C3F90000	504	C3F90504	32-bit
SIUL	IFMC0	32	C3F90000	1000	C3F91000	32-bit
SIUL	IFMC1	32	C3F90000	1004	C3F91004	32-bit
SIUL	IFMC2	32	C3F90000	1008	C3F91008	32-bit
SIUL	IFMC3	32	C3F90000	100C	C3F9100C	32-bit
SIUL	IFMC4	32	C3F90000	1010	C3F91010	32-bit
SIUL	IFMC5	32	C3F90000	1014	C3F91014	32-bit
SIUL	IFMC6	32	C3F90000	1018	C3F91018	32-bit
SIUL	IFMC7	32	C3F90000	101C	C3F9101C	32-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
SIUL	IFMC8	32	C3F90000	1020	C3F91020	32-bit
SIUL	IFMC9	32	C3F90000	1024	C3F91024	32-bit
SIUL	IFMC10	32	C3F90000	1028	C3F91028	32-bit
SIUL	IFMC11	32	C3F90000	102C	C3F9102C	32-bit
SIUL	IFMC12	32	C3F90000	1030	C3F91030	32-bit
SIUL	IFMC13	32	C3F90000	1034	C3F91034	32-bit
SIUL	IFMC14	32	C3F90000	1038	C3F91038	32-bit
SIUL	IFMC15	32	C3F90000	103C	C3F9103C	32-bit
SIUL	IFMC16	32	C3F90000	1040	C3F91040	32-bit
SIUL	IFMC17	32	C3F90000	1044	C3F91044	32-bit
SIUL	IFMC18	32	C3F90000	1048	C3F91048	32-bit
SIUL	IFMC19	32	C3F90000	104C	C3F9104C	32-bit
SIUL	IFMC20	32	C3F90000	1050	C3F91050	32-bit
SIUL	IFMC21	32	C3F90000	1054	C3F91054	32-bit
SIUL	IFMC22	32	C3F90000	1058	C3F91058	32-bit
SIUL	IFMC23	32	C3F90000	105C	C3F9105C	32-bit
SIUL	IFMC24	32	C3F90000	1060	C3F91060	32-bit
SIUL	IFMC25	32	C3F90000	1064	C3F91064	32-bit
SIUL	IFMC26	32	C3F90000	1068	C3F91068	32-bit
SIUL	IFMC27	32	C3F90000	106C	C3F9106C	32-bit
SIUL	IFMC28	32	C3F90000	1070	C3F91070	32-bit
SIUL	IFMC29	32	C3F90000	1074	C3F91074	32-bit
SIUL	IFMC30	32	C3F90000	1078	C3F91078	32-bit
SIUL	IFMC31	32	C3F90000	107C	C3F9107C	32-bit
SIUL	IFCP	32	C3F90000	1080	C3F91080	32-bit
MC Mode Entry						
MC ME	ME_ME	32	C3FDC000	008	C3FDC008	32-bit
MC ME	ME_IM	32	C3FDC000	010	C3FDC010	32-bit
MC ME	ME_TEST_MC	32	C3FDC000	024	C3FDC024	16-bit
MC ME	ME_SAFE_MC	32	C3FDC000	028	C3FDC028	16-bit
MC ME	ME_DRUN_MC	32	C3FDC000	02C	C3FDC02C	16-bit
MC ME	ME_RUN0_MC	32	C3FDC000	030	C3FDC030	16-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
MC ME	ME_RUN1_MC	32	C3FDC000	034	C3FDC034	16-bit
MC ME	ME_RUN2_MC	32	C3FDC000	038	C3FDC038	16-bit
MC ME	ME_RUN3_MC	32	C3FDC000	03C	C3FDC03C	16-bit
MC ME	ME_HALTO_MC	32	C3FDC000	040	C3FDC040	16-bit
MC ME	ME_STOP0_MC	32	C3FDC000	048	C3FDC048	16-bit
MC ME	ME_RUN_PC0	32	C3FDC000	080	C3FDC080	32-bit
MC ME	ME_RUN_PC1	32	C3FDC000	084	C3FDC084	32-bit
MC ME	ME_RUN_PC2	32	C3FDC000	088	C3FDC088	32-bit
MC ME	ME_RUN_PC3	32	C3FDC000	08C	C3FDC08C	32-bit
MC ME	ME_RUN_PC4	32	C3FDC000	090	C3FDC090	32-bit
MC ME	ME_RUN_PC5	32	C3FDC000	094	C3FDC094	32-bit
MC ME	ME_RUN_PC6	32	C3FDC000	098	C3FDC098	32-bit
MC ME	ME_RUN_PC7	32	C3FDC000	09C	C3FDC09C	32-bit
MC ME	ME_LP_PC0	32	C3FDC000	0A0	C3FDC0A0	32-bit
MC ME	ME_LP_PC1	32	C3FDC000	0A4	C3FDC0A4	32-bit
MC ME	ME_LP_PC2	32	C3FDC000	0A8	C3FDC0A8	32-bit
MC ME	ME_LP_PC3	32	C3FDC000	0AC	C3FDC0AC	32-bit
MC ME	ME_LP_PC4	32	C3FDC000	0B0	C3FDC0B0	32-bit
MC ME	ME_LP_PC5	32	C3FDC000	0B4	C3FDC0B4	32-bit
MC ME	ME_LP_PC6	32	C3FDC000	0B8	C3FDC0B8	32-bit
MC ME	ME_LP_PC7	32	C3FDC000	0BC	C3FDC0BC	32-bit
MC ME	ME_PCTL[4..7]	8	C3FDC000	0C4	C3FDC0C4	8-bit
MC ME	ME_PCTL[16..19]	8	C3FDC000	0D0	C3FDC0D0	8-bit
MC ME	ME_PCTL[24..27]	8	C3FDC000	0D8	C3FDC0D8	8-bit
MC ME	ME_PCTL[32..35]	8	C3FDC000	0E0	C3FDC0E0	8-bit
MC ME	ME_PCTL[36..39]	8	C3FDC000	0E4	C3FDC0E4	8-bit
MC ME	ME_PCTL[40..43]	8	C3FDC000	0E8	C3FDC0E8	8-bit
MC ME	ME_PCTL[48..51]	8	C3FDC000	0F0	C3FDC0F0	8-bit
MC ME	ME_PCTL[84..87]	8	C3FDC000	114	C3FDC114	8-bit
MC ME	ME_PCTL[92]	8	C3FDC000	11C	C3FDC11C	8-bit
XOSC, IRC_OSC, FM PLL 0, CMU 0, MC Clock Generation Module						
XOSC	OSC_CTL	32	C3FE0000	000	C3FE0000	32-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
IRC_OSC	RC_CTL	32	C3FE0060	000	C3FE0060	32-bit
FMPLL 0	CR	32	C3FE00A0	000	C3FE00A0	32-bit
FMPLL 0	MR	32	C3FE00A0	004	C3FE00A4	32-bit
CMU 0	CMU_CSR	32	C3FE0100	000	C3FE0100	8-bit
MC CGM	CGM_OC_EN	8	C3FE0000	370	C3FE0370	8-bit
MC CGM	CGM_OCDS_SC	8	C3FE0000	374	C3FE0374	8-bit
MC RGM	RGM_FERD	16	C3FE4000	004	C3FE4004	16-bit
MC RGM	RGM_FEAR	16	C3FE4000	010	C3FE4010	16-bit
MC RGM	RGM_FESS	16	C3FE4000	018	C3FE4018	16-bit
MC RGM	RGM_FBRE	16	C3FE4000	01C	C3FE401C	16-bit
PIT_RTI						
PIT	PIT_PITMCR	32	C3FF0000	000	C3FF0000	32-bit
PIT	PIT_LDVAL0	32	C3FF0000	100	C3FF0100	32-bit
PITI	PIT_TCTRL0	32	C3FF0000	108	C3FF0108	32-bit
PIT	PIT_TFLG0	32	C3FF0000	10C	C3FF010C	32-bit
PIT	PIT_LDVAL1	32	C3FF0000	110	C3FF0110	32-bit
PIT	PIT_TCTRL1	32	C3FF0000	118	C3FF0118	32-bit
PIT	PIT_TFLG1	32	C3FF0000	11C	C3FF011C	32-bit
PIT	PIT_LDVAL2	32	C3FF0000	120	C3FF0120	32-bit
PIT	PIT_TCTRL2	32	C3FF0000	128	C3FF0128	32-bit
PIT	PIT_TFLG2	32	C3FF0000	12C	C3FF012C	32-bit
PIT	PIT_LDVAL3	32	C3FF0000	130	C3FF0130	32-bit
PIT	PIT_TCTRL3	32	C3FF0000	138	C3FF0138	32-bit
PIT	PIT_TFLG3	32	C3FF0000	13C	C3FF013C	32-bit
ADC 0						
ADC 0	MCR	32	FFE00000	000	FFE00000	32-bit
SSCM						
SSCM	ERROR	16	C3FD8000	006	C3FD8006	32-bit
SSCM	DEBUGPORT	16	C3FD8000	0008	C3FD8008	32-bit
eTimer 0						
eTimer 0	CH0_CTRL	16	FFE18000	00E	FFE1800E	16-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
eTimer 0	CH0_CTRL2	16	FFE18000	010	FFE18010	16-bit
eTimer 0	CH0_CTRL3	16	FFE18000	012	FFE18012	16-bit
eTimer 0	CH0_CCCTRL	16	FFE18000	01C	FFE1801C	16-bit
eTimer 0	CH1_CTRL	16	FFE18000	02E	FFE1802E	16-bit
eTimer 0	CH1_CTRL2	16	FFE18000	030	FFE18030	16-bit
eTimer 0	CH1_CTRL3	16	FFE18000	032	FFE18032	16-bit
eTimer 0	CH1_CCCTRL	16	FFE18000	03C	FFE1803C	16-bit
eTimer 0	CH2_CTRL	16	FFE18000	04E	FFE1804E	16-bit
eTimer 0	CH2_CTRL2	16	FFE18000	050	FFE18050	16-bit
eTimer 0	CH2_CTRL3	16	FFE18000	052	FFE18052	16-bit
eTimer 0	CH2_CCCTRL	16	FFE18000	05C	FFE1805C	16-bit
eTimer 0	CH3_CTRL	16	FFE18000	06E	FFE1806E	16-bit
eTimer 0	CH3_CTRL2	16	FFE18000	070	FFE18070	16-bit
eTimer 0	CH3_CTRL3	16	FFE18000	072	FFE18072	16-bit
eTimer 0	CH3_CCCTRL	16	FFE18000	07C	FFE1807C	16-bit
eTimer 0	CH4_CTRL	16	FFE18000	08E	FFE1808E	16-bit
eTimer 0	CH4_CTRL2	16	FFE18000	090	FFE18090	16-bit
eTimer 0	CH4_CTRL3	16	FFE18000	092	FFE18092	16-bit
eTimer 0	CH4_CCCTRL	16	FFE18000	09C	FFE1809C	16-bit
eTimer 0	CH5_CTRL	16	FFE18000	0AE	FFE180AE	16-bit
eTimer 0	CH5_CTRL2	16	FFE18000	0B0	FFE180B0	16-bit
eTimer 0	CH5_CTRL3	16	FFE18000	0B2	FFE180B2	16-bit
eTimer 0	CH5_CCCTRL	16	FFE18000	0BC	FFE180BC	16-bit
IIC 0						
IIC 0	IICA	8	FFFA8000	000	FFFA8000	8-bit
IIC 0	IICF	8	FFFA8000	001	FFFA8001	8-bit
IIC 0	IICCR	8	FFFA8000	005	FFFA8005	8-bit
IIC 1						
IIC 1	IICA	8	FFFAC000	000	FFFAC000	8-bit
IIC 1	IICF	8	FFFAC000	001	FFFAC001	8-bit
IIC 1	IICCR	8	FFFAC000	005	FFFAC005	8-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
LINFLEX 0						
LINFlex0	LINCR1	32	FFE40000	000	FFE40000	32-bit
LINFlex0	LINIER	32	FFE40000	004	FFE40004	32-bit
LINFlex0	UARTCR	32	FFE40000	010	FFE40010	32-bit
LINFlex0	LINTCSR	32	FFE40000	018	FFE40018	32-bit
LINFlex0	LINOCR	32	FFE40000	01C	FFE4001C	32-bit
LINFlex0	LINTOCR	32	FFE40000	020	FFE40020	32-bit
LINFlex0	LINFBRR	32	FFE40000	024	FFE40024	32-bit
LINFlex0	LINCR2	32	FFE40000	030	FFE40030	32-bit
LINFlex0	BIDR	32	FFE40000	034	FFE40034	32-bit
LINFlex0	IFER	32	FFE40000	040	FFE40040	32-bit
LINFlex0	IFMR	32	FFE40000	048	FFE40048	32-bit
LINFlex0	IFCR0	32	FFE40000	04C	FFE4004C	32-bit
LINFlex0	IFCR1	32	FFE40000	050	FFE40050	32-bit
LINFlex0	IFCR2	32	FFE40000	054	FFE40054	32-bit
LINFlex0	IFCR3	32	FFE40000	058	FFE40058	32-bit
LINFlex0	IFCR4	32	FFE40000	05C	FFE4005C	32-bit
LINFlex0	IFCR5	32	FFE40000	060	FFE40060	32-bit
LINFlex0	IFCR6	32	FFE40000	064	FFE40064	32-bit
LINFlex0	IFCR7	32	FFE40000	068	FFE40068	32-bit
LINFlex0	IFCR8	32	FFE40000	06C	FFE4006C	32-bit
LINFlex0	IFCR9	32	FFE40000	070	FFE40070	32-bit
LINFlex0	IFCR10	32	FFE40000	074	FFE40074	32-bit
LINFlex0	IFCR11	32	FFE40000	078	FFE40078	32-bit
LINFlex0	IFCR12	32	FFE40000	07C	FFE4007C	32-bit
LINFlex0	IFCR13	32	FFE40000	080	FFE40080	32-bit
LINFlex0	IFCR14	32	FFE40000	084	FFE40084	32-bit
LINFlex0	IFCR15	32	FFE40000	088	FFE40088	32-bit
LINFLEX 1						
LINFlex1	LINCR1	32	FFE44000	000	FFE44000	32-bit
LINFlex1	LINIER	32	FFE44000	004	FFE44004	32-bit
LINFlex1	UARTCR	32	FFE44000	010	FFE44010	32-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
LINFlex1	LINTCSR	32	FFE44000	018	FFE44018	32-bit
LINFlex1	LINOCR	32	FFE44000	01C	FFE4401C	32-bit
LINFlex1	LINTOCR	32	FFE44000	020	FFE44020	32-bit
LINFlex1	LINFBRR	32	FFE44000	024	FFE44024	32-bit
LINFlex1	LINCR2	32	FFE44000	030	FFE44030	32-bit
LINFlex1	BIDR	32	FFE44000	034	FFE44034	32-bit
CRC						
CRC	CFG0	32	FFE68000	000	FFE68000	32-bit
CRC	CFG1	32	FFE68000	010	FFE68010	32-bit
DSPI 0						
DSPI 0	DSPI_MCR	32	FFF90000	000	FFF90000	32-bit
DSPI 0	DSPI_TCR	32	FFF90000	008	FFF90008	32-bit
DSPI 0	DSPI_CTAR0	32	FFF90000	00C	FFF9000C	32-bit
DSPI 0	DSPI_CTAR1	32	FFF90000	010	FFF90010	32-bit
DSPI 0	DSPI_CTAR2	32	FFF90000	014	FFF90014	32-bit
DSPI 0	DSPI_CTAR3	32	FFF90000	018	FFF90018	32-bit
DSPI 0	DSPI_CTAR4	32	FFF90000	01C	FFF9001C	32-bit
DSPI 0	DSPI_CTAR5	32	FFF90000	020	FFF90020	32-bit
DSPI 0	DSPI_CTAR6	32	FFF90000	024	FFF90024	32-bit
DSPI 0	DSPI_CTAR7	32	FFF90000	028	FFF90028	32-bit
DSPI 0	DSPI_RSER	32	FFF90000	030	FFF90030	32-bit
DSPI 1						
DSPI 1	DSPI_MCR	32	FFF94000	000	FFF94000	32-bit
DSPI 1	DSPI_TCR	32	FFF94000	008	FFF94008	32-bit
DSPI 1	DSPI_CTAR0	32	FFF94000	00C	FFF9400C	32-bit
DSPI 1	DSPI_CTAR1	32	FFF94000	010	FFF94010	32-bit
DSPI 1	DSPI_CTAR2	32	FFF94000	014	FFF94014	32-bit
DSPI 1	DSPI_CTAR3	32	FFF94000	018	FFF94018	32-bit
DSPI 1	DSPI_CTAR4	32	FFF90000	01C	FFF9001C	32-bit
DSPI 1	DSPI_CTAR5	32	FFF90000	020	FFF90020	32-bit
DSPI 1	DSPI_CTAR6	32	FFF90000	024	FFF90024	32-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
DSPI 1	DSPI_CTAR7	32	FFF90000	028	FFF90028	32-bit
DSPI 1	DSPI_RSER	32	FFF94000	030	FFF94030	32-bit
DSPI 2						
DSPI 2	DSPI_MCR	32	FFF98000	000	FFF98000	32-bit
DSPI 2	DSPI_TCR	32	FFF98000	008	FFF98008	32-bit
DSPI 2	DSPI_CTAR0	32	FFF98000	00C	FFF9800C	32-bit
DSPI 2	DSPI_CTAR1	32	FFF98000	010	FFF98010	32-bit
DSPI 2	DSPI_CTAR2	32	FFF98000	014	FFF98014	32-bit
DSPI 2	DSPI_CTAR3	32	FFF98000	018	FFF98018	32-bit
DSPI 2	DSPI_CTAR4	32	FFF90000	01C	FFF9001C	32-bit
DSPI 2	DSPI_CTAR5	32	FFF90000	020	FFF90020	32-bit
DSPI 2	DSPI_CTAR6	32	FFF90000	024	FFF90024	32-bit
DSPI 2	DSPI_CTAR7	32	FFF90000	028	FFF90028	32-bit
DSPI 2	DSPI_RSER	32	FFF98000	030	FFF98030	32-bit
FlexCAN						
FlexCAN	CANx_MCR	32	FFFC0000	000	FFFC0000	32-bit
FlexCAN	CANx_CTRL	32	FFFC0000	004	FFFC0004	32-bit
FlexCAN	CANx_RXGMASK	32	FFFC0000	010	FFFC0010	32-bit
FlexCAN	CANx_RX14MASK	32	FFFC0000	014	FFFC0014	32-bit
FlexCAN	CANx_RX15MASK	32	FFFC0000	018	FFFC0018	32-bit
FlexCAN	CANx_IMASK2	32	FFFC0000	024	FFFC0024	32-bit
FlexCAN	CANx_IMASK	32	FFFC0000	028	FFFC0028	32-bit
DMA Channel Mux						
DMACH–Mux	CHCONFIG0	8	FFFDC000	000	FFFDC000	8-bit
DMACH–Mux	CHCONFIG1	8	FFFDC000	001	FFFDC001	8-bit
DMACH–Mux	CHCONFIG2	8	FFFDC000	002	FFFDC002	8-bit
DMACH–Mux	CHCONFIG3	8	FFFDC000	003	FFFDC003	8-bit
DMACH–Mux	CHCONFIG4	8	FFFDC000	004	FFFDC004	8-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
DMACH–Mux	CHCONFIG5	8	FFFDC000	005	FFFDC005	8-bit
DMACH–Mux	CHCONFIG6	8	FFFDC000	006	FFFDC006	8-bit
DMACH–Mux	CHCONFIG7	8	FFFDC000	007	FFFDC007	8-bit
DMACH–Mux	CHCONFIG8	8	FFFDC000	008	FFFDC008	8-bit
DMACH–Mux	CHCONFIG9	8	FFFDC000	009	FFFDC009	8-bit
DMACH–Mux	CHCONFIG10	8	FFFDC000	00A	FFFDC00A	8-bit
DMACH–Mux	CHCONFIG11	8	FFFDC000	00B	FFFDC00B	8-bit
DMACH–Mux	CHCONFIG12	8	FFFDC000	00C	FFFDC00C	8-bit
DMACH–Mux	CHCONFIG13	8	FFFDC000	00D	FFFDC00D	8-bit
DMACH–Mux	CHCONFIG14	8	FFFDC000	00E	FFFDC00E	8-bit
DMACH–Mux	CHCONFIG15	8	FFFDC000	00F	FFFDC00F	8-bit
SAI 0						
SAI 0	STCR2	32	FFFD8000	008	FFFD8008	32-bit
SAI 0	STCR3	16	FFFD8000	00C	FFFD800C	16-bit
SAI 0	STCR4	32	FFFD8000	010	FFFD8010	32-bit
SAI 0	STCR5	32	FFFD8000	014	FFFD8014	32-bit
SAI 0	SRCR2	32	FFFD8000	088	FFFD8088	32-bit
SAI 0	SRCR3	16	FFFD8000	08C	FFFD808C	16-bit
SAI 0	SRCR4	32	FFFD8000	090	FFFD8090	32-bit
SAI 0	SRCR5	32	FFFD8000	094	FFFD8094	32-bit
SAI 1						
SAI 1	STCR2	32	FFFF0000	008	FFFF0008	32-bit
SAI 1	STCR3	16	FFFF0000	00C	FFFF000C	16-bit
SAI 1	STCR4	32	FFFF0000	010	FFFF0010	32-bit
SAI 1	STCR5	32	FFFF0000	014	FFFF0014	32-bit

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
SAI 1	SRCR2	32	FFFF0000	088	FFFF0088	32-bit
SAI 1	SRCR3	16	FFFF0000	08C	FFFF008C	16-bit
SAI 1	SRCR4	32	FFFF0000	090	FFFF0090	32-bit
SAI 1	SRCR5	32	FFFF0000	094	FFFF0094	32-bit
SAI 2						
SAI 2	STCR2	32	FFFF4000	008	FFFF4008	32-bit
SAI 2	STCR3	16	FFFF4000	00C	FFFF400C	16-bit
SAI 2	STCR4	32	FFFF4000	010	FFFF4010	32-bit
SAI 2	STCR5	32	FFFF4000	014	FFFF4014	32-bit
SAI 2	SRCR2	32	FFFF4000	088	FFFF4088	32-bit
SAI 2	SRCR3	16	FFFF4000	08C	FFFF408C	16-bit
SAI 2	SRCR4	32	FFFF4000	090	FFFF4090	32-bit
SAI 2	SRCR5	32	FFFF4000	094	FFFF4094	32-bit
FEC						
FEC	MSCR	32	FFF4C000	044	FFF4C044	32-bit
FEC	PALR	32	FFF4C000	0E4	FFF4C0E4	32-bit
FEC	PAUR	16	FFF4C000	0E8	FFF4C0E8	16-bit
PTP						
PTP	TSPDR1	32	FFE74000	000	FFE74000	32
PTP	TSPDR2	32	FFE74000	004	FFE74004	32
PTP	TSPDR3	32	FFE74000	008	FFE74008	32
PTP	TSPDR4	32	FFE74000	00C	FFE7400C	32
PTP	TSPOV	32	FFE74000	010	FFE74010	32
PTP	TSMR	32	FFE74000	014	FFE74014	32
PTP	TMR_PEMASK	32	FFE74000	01C	FFE7401C	32
PTP	TSPDR5	32	FFE74000	060	FFE74060	32
PTP	TSPDR6	32	FFE74000	064	FFE74064	32
PTP	TSPDR7	32	FFE74000	068	FFE74068	32
CE_RTC						
CE_RTC	TMR_CTRL	32	FFE78000	000	FFE78000	32
CE_RTC	TMR_TEMASK	32	FFE78000	008	FFE78008	32

Table 540. Registers under protection

Module	Register	Register size	Module Base	Register Offset	Absolute address	Protect size
CE_RTC	TMR_PRSC	32	FFE78000	01C	FFE7801C	32
CE_RTC	TMR_ALARM1_L	32	FFE78000	028	FFE78028	32
CE_RTC	TMR_ALARM1_H	32	FFE78000	02C	FFE7802C	32
CE_RTC	TMR_ALARM2_L	32	FFE78000	030	FFE78030	32
CE_RTC	TMR_ALARM2_H	32	FFE78000	034	FFE78034	32
CE_RTC	TMR_FIPER1	32	FFE78000	038	FFE78038	32
CE_RTC	TMR_FIPER2	32	FFE78000	03C	FFE7803C	32
CE_RTC	TMR_FIPER3	32	FFE78000	040	FFE78040	32
CE_RTC	TMR_FSV_L	32	FFE78000	054	FFE78054	32
CE_RTC	TMR_FSV_H	32	FFE78000	058	FFE78058	32
VID 0						
VID 0	STC	32	0xFFFF8000	000	FFFF8000	8-bit (15:8)
VID 0	PSZ	32	0xFFFF8000	004	FFFF8004	14-bit (13:0)
VID 0	CFGM	32	0xFFFF8000	034	FFFF8034	11-bit (10:0)

Chapter 40

Temperature Sensor (TSENS)

40.1 Introduction

The TSENS module provides an analog voltage that is proportional to the internal temperature of the MPC5604E. This voltage can be sampled by the ADC and converted to an actual temperature by using factory-programmed calibration constants.

Figure 535 shows a block diagram of the TSENS module. The module operates on the principles of a basic silicon bandgap sensor that measures the difference in base-emitter voltages of multiple transistors, which can be related to the absolute temperature of the device.

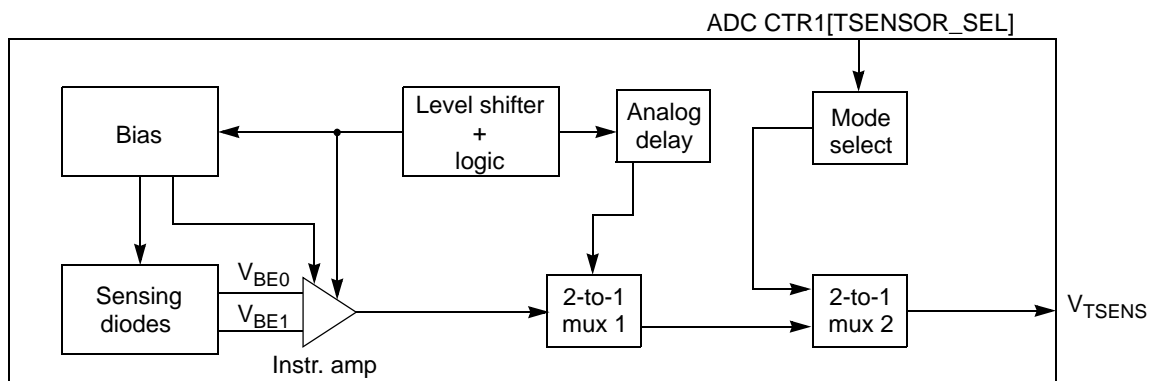


Figure 535. TSENS block diagram

40.2 Features

The TSENS has the following features:

- Temperature monitoring range: -40 to 150 °C
- Sensitivity: Approximately 5.14 mV/°C

40.3 Signals

The TSENS does not use any input signals external to the device.

The TSENS outputs one analog signal, V_{TSENS} . Depending on the TSENS mode of operation, V_{TSENS} is proportional or inversely proportional to the device temperature. The V_{TSENS} signal is connected to ADC_0 Channel 4. See Chapter 27, "Analog-to-Digital Converter (ADC)", for more information on how ADC channels are connected to V_{TSENS} and to other modules.

40.4 Memory Map and Register Description

This section provides a detailed description of all registers accessible in the TSENS module.

40.4.1 Memory Map

The TSENS module has no memory-mapped configuration or status registers. However, data can be read from Flash memory. See [Chapter 18, "Flash Memory"](#).

40.5 Modes of operation

The TSENS has two modes of operation:

- Proportional to absolute temperature (PTAT) — V_{TSENS} increases linearly with increasing temperature
- Complementary to absolute temperature (CTAT) — V_{TSENS} decreases linearly with increasing temperature

The mode of operation is controlled by the CTR1[TSENSOR_SEL] field in the ADC.

40.6 Obtaining the device temperature using TSENS

To obtain the device temperature using TSENS, do the following:

- Extract the necessary TSENS calibration constants from the MPC5604E test flash memory (see [Section 40.6.1, "TSENS calibration constants"](#))
- Measure V_{TSENS} in PTAT and CTAT modes

NOTE

In the PTAT mode, ADC sampling time should be 1 μ s. In the CTAT mode, ADC sampling time should be 2 μ s.

- Calculate the device temperature using the equations in [Section 40.6.2, "Equations for converting TSENS voltage to device temperature"](#)

The need for the measurements in two different modes is driven by the fact that although the TSENS output is linear in either mode, the intercept of the V_{TSENS} -T plot varies significantly based on the ADC reference voltage. Performing the two measurements allows the resulting equations to be independent of this reference voltage.

40.6.1 TSENS calibration constants

The equations needed to convert V_{TSENS} to a device temperature depend on four calibration constants as described in [Table 541](#). These constants are determined during factory testing of the MPC5604E and stored in the MPC5604E test flash memory.

Table 541. TSENS calibration constants

Constant	Test flash memory address	Description
P _{cold}	0x0000	Tsens_code ¹ (binary value) measured at -40 °C
P _{hot}	0x0002	Tsens_code(binary value) measured at 150°C
C _{cold}	0x0004	Tsens_code(binary value) measured at -40°C
C _{hot}	0x0006	Tsens_code(binary value) measured at 150 °C

¹ Tsens_code is the temperature sensor voltage sampled by the ADC and converted to a 10-bit value

$$P_{cold/hot}(C_{cold/hot}) = (T_{sens_code}/1024)*V_{ref}$$

V_{ref} is ADC reference voltage value 3.3V

40.6.2 Equations for converting TSENS voltage to device temperature

In the equations below:

- V_P is the V_{TSENS} output in PTAT mode in V
- V_C is the V_{TSENS} output in CTAT mode in V
- P_n and C_n are the calibration constants described in [Section 40.6.1, “TSENS calibration constants”](#)
- T is the device temperature in °C

Calculate the device temperature as:

$$T = -40 + 190 \frac{(V_p - V_c - P_{cold} + C_{cold})}{(P_{hot} - C_{hot} - P_{cold} + C_{cold})}$$

Eqn. 1

THIS PAGE INTENTIONALLY BLANK

Chapter 41

JTAG Controller (JTAGC)

41.1 Introduction

Figure 536 is a block diagram of the JTAG Controller (JTAGC) block.

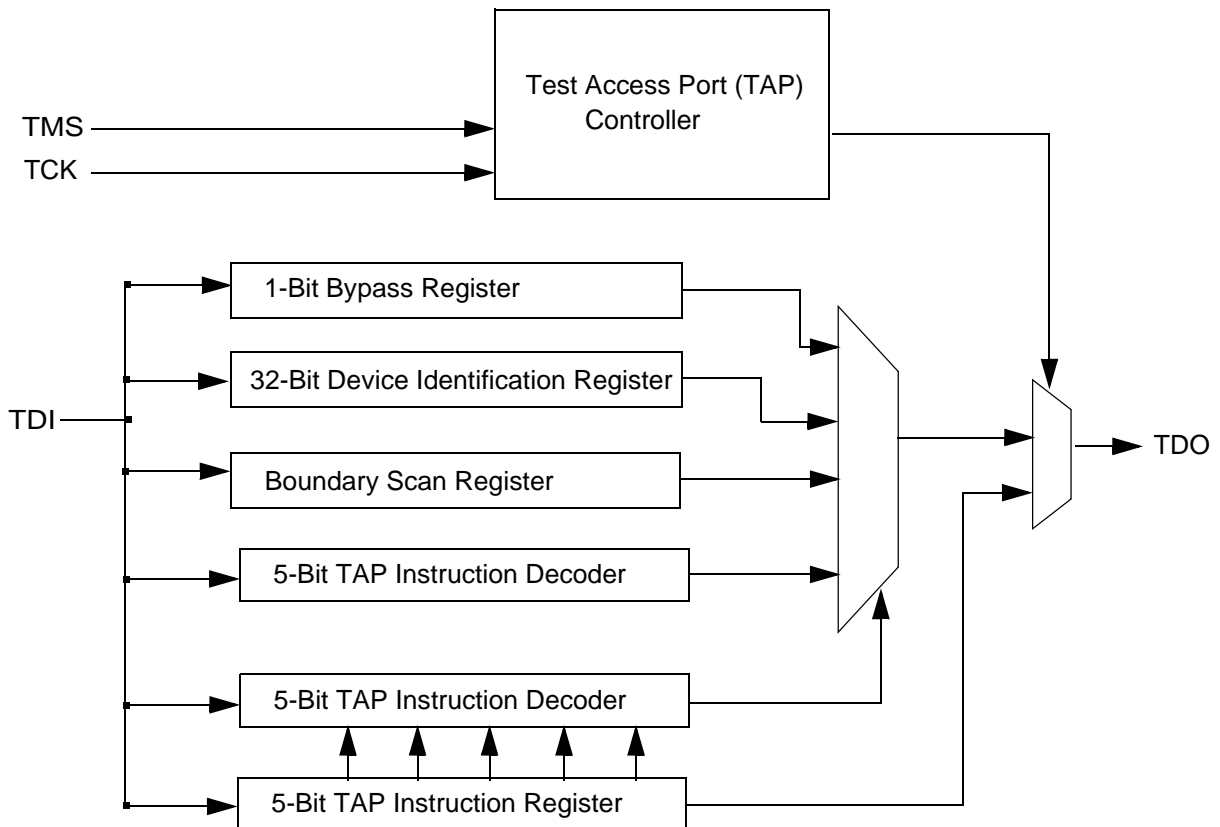


Figure 536. JTAG STL (IEEE 1149.1) Block Diagram

41.1.1 Overview

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

41.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
 - 4 pins (TDI, TMS, TCK, and TDO)
- A 5-bit instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to [Table 543](#) for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_TAP_x instructions.
- Test data registers: a bypass register, a boundary scan register, and a device identification register.
- A TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

41.1.3 Modes of Operation

The JTAGC block uses power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

41.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, , or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

41.1.3.2 IEEE 1149.1-2001 Defined Test Modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in [Section 41.4.4, JTAGC Block Instructions](#).

41.1.3.3 Bypass Mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

41.2 External Signal Description

41.2.1 Overview

The JTAGC consists of 5 signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in [Table 542](#).

Table 542. JTAG Signal Properties

Name	I/O	Function	Reset State	Pull ¹
TCK	Input	Test Clock	-	Up
TDI	Input	Test Data In	-	Up
TDO	Output	Test Data Out	High Z ²	-
TMS	Input	Test Mode Select	-	Up

- ¹ The pull is not implemented in this block. Pullup/pulldown devices are implemented in the pads.
² TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

NOTE

TCK frequency should always be less than the system clock (Sys_clk) frequency.

41.2.2 Detailed Signal Descriptions

This section describes each of the signals listed in [Table 542](#) in more detail.

41.2.2.1 TCK - Test Clock Input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

41.2.2.2 TDI - Test Data Input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

41.2.2.3 TDO - Test Data Output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in [Section 41.4.3, TAP Controller State Machine](#). The TDO output of this

block is clocked on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

41.2.2.4 TMS - Test Mode Select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

41.3 Register Definition

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

41.3.1 Register Descriptions

The JTAGC block registers are described in this section.

41.3.1.1 Instruction Register

The JTAGC block uses a 5-bit instruction register as shown in [Table 537](#). The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 0b10101, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

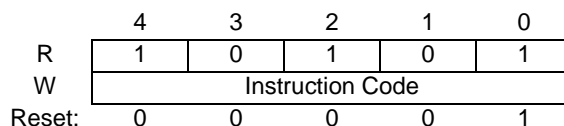


Figure 537. 5-Bit Instruction Register

41.3.1.2 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

41.3.1.3 Device Identification Register

The device identification register, shown in Figure 538, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state. The part revision number (PRN) and part identification number (PIN) fields are system plugs, and the manufacturer identity code (MIC) is a constant value assigned to the manufacturer by the JEDEC.

The shift register LSB is forced to logic 1 on the rising edge of TCK following entry into the Capture-DR state. Therefore, the first bit to be shifted out after selecting the IDCODE register is always a logic 1. The remaining 31 bits are forced to the value of the device identification register on the rising edge of TCK following entry into the Capture-DR state.

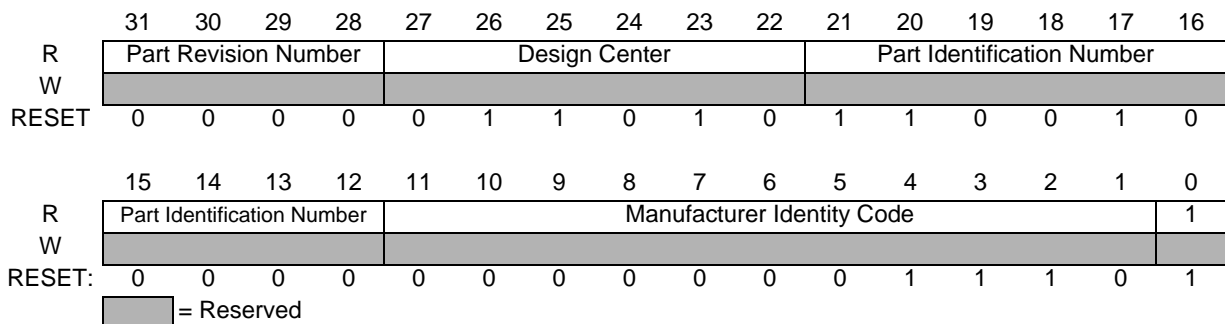


Figure 538. Device Identification Register

PRN — Part Revision Number

Bits [31:28] contain the revision number of the part.

DC — Design Center

Bits [27:22] indicate the design center.

PIN — Part Identification Number

Bits [21:12] contain the part number of the device.

MIC — Manufacturer Identity Code

Bits [11:1] contain the reduced Joint Electron Device Engineering Council (JEDEC) ID.

Bit [0] — IDCODE Register ID

Bit [0] identifies this register as the device identification register and not the bypass register

41.3.1.4 CENSOR_CTRL Register

The CENSOR_CTRL register is a 64-bit shift register path from TDI to TDO selected when the ENABLE_CENSOR_CTRL instruction is active. The default reset value of the CENSOR_CTRL register

is 64'b0 . The CENSOR_CTRL register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_CENSOR_CTRL instruction is executed, the register value will remain valid until a JTAG reset occurs.

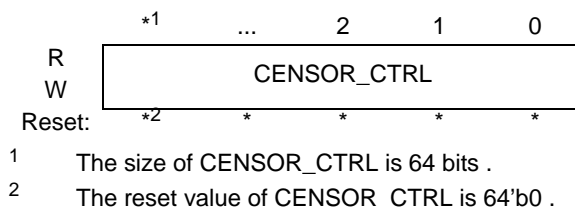


Figure 539. CENSOR_CTRL Register

CENSOR_CTRL - Censorship Control

The CENSOR_CTRL bits are used to control chiptop censorship functions.

41.3.1.5 Boundary Scan Register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in [Section 41.4.5, Boundary Scan](#). The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

41.4 Functional Description

41.4.1 JTAGC Reset Configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

41.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. For more detail on TAP sharing via JTAGC instructions refer to [Section 41.4.4.7, ACCESS_AUX_TAP_x Instructions](#).

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in [Figure 540](#). This applies for the instruction register, test data registers, and the bypass register.

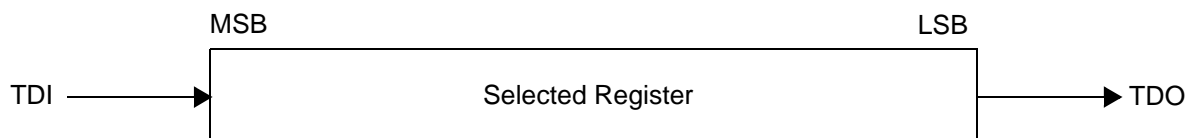
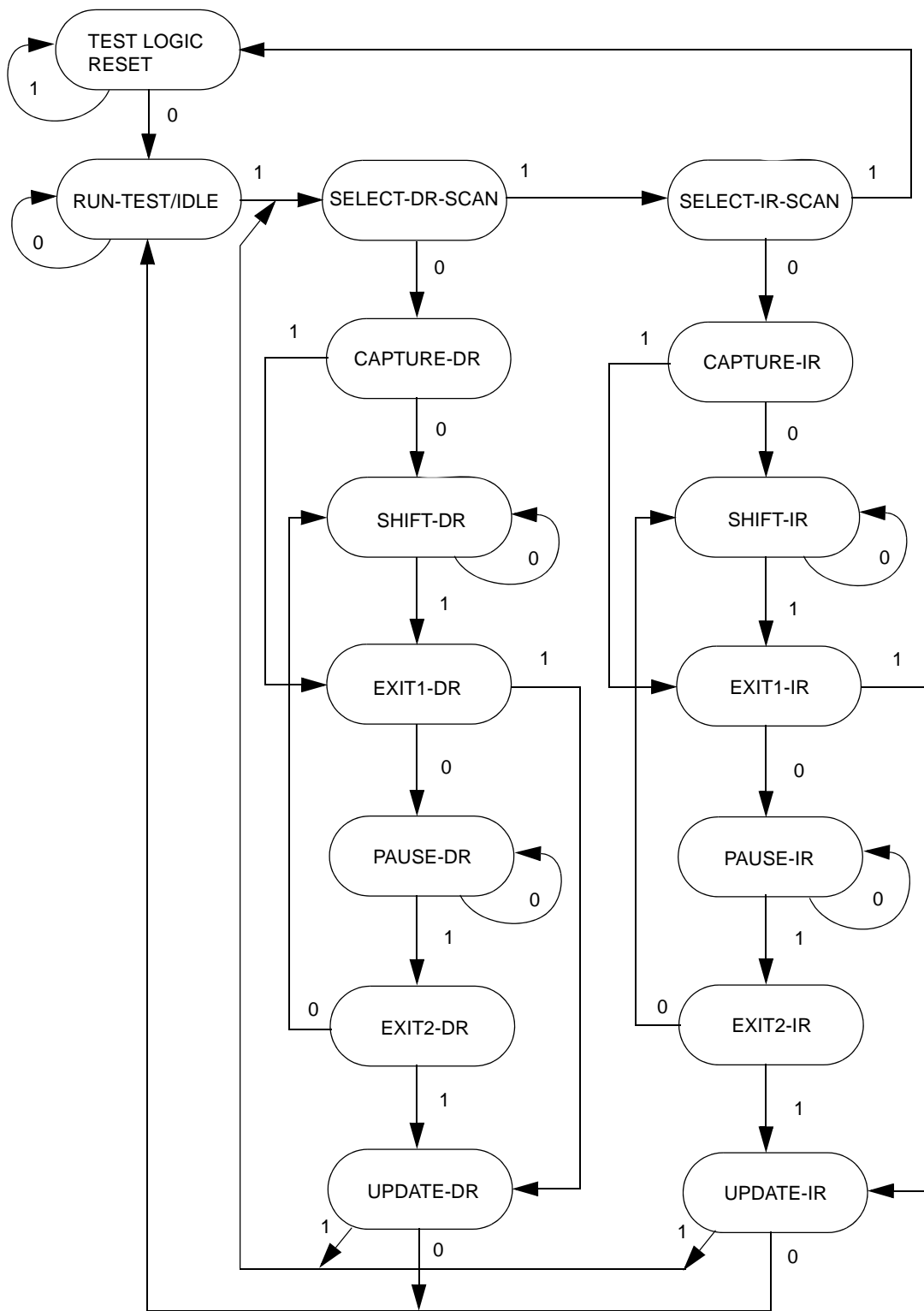


Figure 540. Shifting Data Through a Register

41.4.3 TAP Controller State Machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. [Figure 541](#) shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As [Figure 541](#) shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 541. IEEE 1149.1-2001 TAP Controller Finite State Machine

41.4.3.1 Enabling the TAP Controller

The JTAGC TAP controller is enabled by placing the JTAGC in reset. The JTAGC block is placed in reset when either power-on reset is asserted, or the TMS input is held high for enough consecutive rising edges of TCK.

41.4.3.2 Selecting an IEEE 1149.1-2001 Register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

41.4.4 JTAGC Block Instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in [Table 543](#). This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 543. JTAG Instructions

Instruction	Code[4:0]	Instruction Summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
ENABLE_CENSOR_CTRL	00111	Selects CENSOR_CTRL register
CLAMP	01100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ACCESS_AUX_TAP_x	10000-11110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below. The number of auxiliary TAP controllers sharing the port is SHARE_CNT
BYPASS	11111	Selects bypass register for data operations
Factory debug reserved	00101, 00110, 01010, 00111	Intended for factory debug only
Reserved ¹	All other opcodes	Decoded to select bypass register

¹ The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

41.4.4.1 IDCODE Instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

41.4.4.2 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

41.4.4.3 SAMPLE Instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

41.4.4.4 EXTEST — External Test Instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

41.4.4.5 ENABLE_CENSOR_CTRL Instruction

The ENABLE_CENSOR_CTRL instruction selects the CENSOR_CTRL register for connection as the shift path between TDI and TDO.

41.4.4.6 CLAMP Instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

41.4.4.7 ACCESS_AUX_TAP_x Instructions

The JTAGC is configurable to allow up to fifteen other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_TAP_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

41.4.4.8 BYPASS Instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

41.4.5 Boundary Scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

41.5 Initialization/Application Information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Enable the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

THIS PAGE IS INTENTIONALLY LEFT BLANK

Appendix A

Revision History of this Document

This appendix describes corrections to the MPC5604E Reference Manual. For convenience, the corrections are grouped by revision. Grammatical and formatting changes are not listed here unless the meaning of something changed.

NOTE

This revision history uses clickable cross-references for ease of navigation. The numbers and titles in each cross-reference are relative to the latest published release.

A.1 Changes between revisions 4 and 5

Table A-1. Changes between revisions 4 and 5

Chapter	Description
Overview	<ul style="list-style-type: none"> In Figure 1., “MPC5604E block diagram”, changed “4+4 channels” to “4+3 channels”. In Figure 4., “MPC5604E video data path”, changed “4+4 channels” to “4+3 channels”. In Figure 5., “Audio to ethernet data path”, changed “4+4 channels” to “4+3 channels”. In Section 1.8, “Chip-level features”, changed from “4 input channels” to “7 input channels: 4 channels routed to the pins, 3 internal connections (temperature sensor, core voltage, IO voltage)”. In Section 1.8.22, “Successive approximation Analog-to-Digital Converter (ADC)”, changed from “4 internal connections” to “3 internal connections”.
Signal Description	<ul style="list-style-type: none"> Updated Table 5., “Supply pins” In Figure 6., “64-pin LQFP pinout (top view)”, updated pin name of pin 47 from VSS to VSS_HV. In Figure 7., “100-pin LQFP pinout (top view)”, updated pin name of pin 74 from VSS to VSS_HV.
Clock Architecture	<p>Previous errata ERR008227 integrated into Reference Manual: Added the following note in Section 4.2, “High-level block diagrams”:</p> <ul style="list-style-type: none"> Software should take care that the clock sources (system clocks and protocol clocks) clock sources selected for a peripheral are not disabled during any mode transition preceding a mode transition to disable the peripheral. For example: In FLEXCAN, when FXOSC is selected for the CAN Engine clock source (FLEXCAN_CTRL[CLK_SRC]), before disabling FLEXCAN peripheral clock, the user must ensure that FXOSC is enabled in the current mode prior to the target mode transition that will disable the FLEXCAN. FXOSC is enabled in the current mode prior to the target mode transition that will disable the FLEXCAN. Software should take care that the clock sources (system clocks and protocol clocks) are enabled for the target mode before initiating the target mode transition to enable the peripheral.

Table A-1. Changes between revisions 4 and 5

Chapter	Description
System Integration Unit Lite (SIUL)	<ul style="list-style-type: none"> Updated the descriptions of PARTNUM[15:0] and MINOR_MASK[3:0] bits in Table 94., “MIDR1 field descriptions”. Updated Figure 98., “MCU ID Register #2 (MIDR2)”. Updated Table 95., “MIDR2 field descriptions”.
Enhanced Direct Memory Access (eDMA)	<ul style="list-style-type: none"> Whole chapter reviewed and significant changes added
Analog to Digital Converter (ADC)	<ul style="list-style-type: none"> In Section 27.2.1, “Features”, updated the entries for channel 5 and channel 6. Also, changed “8 input channels” to “7 input channels”. Updated Figure 331., “ADC Block Diagram”. Removed CDR[7] row from Table 355., “ADC digital registers”. Removed DMA7 bit field from Figure 339., “DMA Channel Select Register (DMAR)”. In Section 27.3.6, “Conversion timing registers CTR[0..1]”, changed “from 4 to 7” to “from “4 to 6”. Removed CH7 bit field from Figure 343., “Normal Conversion Mask Register 0 (NCMR0)”. Removed CH7 bit field from Figure 344., “Injected Conversion Mask Register 0 (JCMR0)”. Removed WSEL_CH7 bit field from Figure 347., “Channel Watchdog Select Register 0 (CWSELR0)”. Removed CWEN7 bit field from Figure 348., “Channel Watchdog Enable Register (CWENR0)”. In Table 364., “DMA Channel Select Register 0 (DMAR0) field descriptions”, removed the row for field 31. Updated Table 368., “Normal Conversion Mask Register 0 (NCMR0) field descriptions” Updated Table 369., “Injected Conversion Mask Register 0 (JCMR0) field descriptions” In Section 27.3.9.1, “Introduction”, changed from “CDR[4..7]” to “CDR[4..6]”. Updated the register name in Section 27.3.9.2, “Channel Data Register (CDR[0..6])” (from earlier name of Channel Data Register (CDR[7])) In Table 371., “CDR[0..6] field descriptions”, updated the description of bit 22:31: Removed “0-95” text from description
Enhanced Motor Control Timer (eTimer)	<ul style="list-style-type: none"> Previous errata ERR006802 integrated into Reference Manual: Added the following paragraph in Section 28.8, “DMA”: When DMA is used to read the eTimer_CAPTn registers and the DMA has completed its programmed number of transfers, then the extra input capture events will set the eTimer_STS[ICFn] bits and also set the internal DMA request signal. While the ICFn bits can be cleared by writing a 1 to their bit positions, the DMA request can only be cleared by the internal DMA done signal. This means that when a new DMA transfer is programmed, the eTimer will request a DMA read with possibly unwanted data. In cases where extra eTimer input capture events might occur, the following procedure can be used to prevent unwanted DMA read requests: Upon completion of the DMA transfer, clear the Timer_INTDMA[ICFnDE] bits. Previous errata ERR006583 integrated into Reference Manual: Added the following paragraph in Section 28.4.4.6, “Hold Register (HOLD)”: In addition, this read only register stores the counter’s value if any of the Compare and Capture Control Registers (CCCTRL) within a module are read. Previous errata ERR006239 integrated into Reference Manual: In Section 28.4.4.11, “Status Register (STS)”, removed “while the counter is enabled” text from description of ICF1 and ICF2 bits.
Temperature Sensor (TSENS)	<ul style="list-style-type: none"> In Table 541, TSENS calibration constants, updated “test flash memory address” of Chot constant to 0x0006.

A.2 Changes between revisions 3 and 4

Table A-2. Changes between revisions 3 and 4

Chapter	Description
Overview	Updated Figure 1 , by replace SIU by SIUL (System Integration Unit Lite)
Signal Description	Updated Table 4 , Pin muxing with GPI only ports.
Flash Memory	Replaced references to ECSM with MCM. In Section 18.2.4.2.1, Platform Flash Configuration Register 0 (PFCR0) , added note, PFCR0[BK0_APC] must equal to PFCR0[BK0_RWSC]. Refer datasheet for correct setting of RWSC. In Section 18.2.4.2.2, Platform Flash Configuration Register 1 (PFCR1) , added note, PFCR1[BK1_APC] must equal to PFCR1[BK1_RWSC]. Refer datasheet for correct setting of RWSC.
Lin Controller (LINFlex)	In Table 320, LINSR field descriptions , added note that LIN state bits (LINS3:0) should be used in the LIN mode only. In Section 25.8.3.1, LIN timeout mode and Section 25.8.3.2, Output compare mode , updated information regarding clearing and setting of the LTOM bit in the LINTCSR to enable timeout and compare mode respectively.
Analog to Digital Converter (ADC)	Updated Section 27.4.1.3, Normal conversion operating modes with note for one shot mode.
Nexus Debug Interface (NDI)	In Table 526, NPC Signal Properties updated pull value for TCK to Up.
Temperature Sensor (TSENS)	In Table 541, TSENS calibration constants , updated calibration constant value. In Section 40.6.2, Equations for converting TSENS voltage to device temperature , updated equations.
JTAG Controller (JTAGC)	In Table 542, JTAG Signal Properties updated pull value for TCK to Up.
Enhanced Direct Memory Access (eDMA)	In Table 227, TCDn 32-bit memory structure , and Figure 184 , updated NBYTES field.

NOTE

This revision history uses clickable cross-references for ease of navigation. The numbers and titles in each cross-reference are relative to the latest published release.

A.3 Changes between revisions 2 and 3

Table A-3. Changes between revisions 2 and 3

Chapter	Description
Overview	Replace SIU by SIUL (System Integration Unit Lite)
Memory Map	In Table 3, Peripheral memory map , moved Ethernet (FEC)0xFFFF4_C000 and Reserved 0xFFFF5_0000 from "AIPS(0)-On Platform Peripherals" to "AIPS(0)-Off Platform Peripherals".

Table A-3. Changes between revisions 2 and 3 (continued)

Chapter	Description
Clock Architecture	Added note in Section 4.7.4.2, "Frequency meter" . Added note in the register description of "Internal RC oscillator (IRC) digital interface" and "External crystal oscillator (XOSC) digital interface" (Section 4.4.3, "Register description" and Section 4.5.3, "Register description")
Mode Entry Module	In Table 36.MC_ME Mode Descriptions, changed the exit from "interrupt event" to "off platform interrupt event"for HALT0. In Section 6.5.2.5, changed the text from "interrupt event" to "off platform interrupt event"for HALT0. In Table 53.MC_ME Resource Control Overview <ul style="list-style-type: none"> • XOSC/FMPLL changed to 'on' in RUN0..3. • Removed PD0 row. • Changed Code and Data flash to normal mode for Halt0 and Stop0 mode.
Interrupt Controller	In Table 75,"Interrupt vectors" , added OFFSET for interrupt sources with IRQ number 157 to 221. In Table 75,"Interrupt vectors" , added table footnote for IRQ and ERR giving the description
System Integration Unit Lite	Modified the description of MAXCNTx field in Table 111, IFMC field descriptions .
Internal Static RAM	Replaced ECSM by MCM
Video Encoder Wrapper	In Table 229, Picture size register fields , changed bit field description of picture_hsize[10:5] from "Number of pixels in a line should be divided by 16 and not 32" to "Number of pixels in a line should be divided by 32." In Table 236, The RC_REGS_SEL control register , added description for LumaTruncH,LumaTruncL,ChromaTruncH, ChromaTruncL. In Table 239, CFG_MODE field description , added bit field decription when the bit is set to 1 for MDRI and to 0 for COMN_DQT,COMB_DHT and DICOM.
Deserial Serial Peripheral Interface	In Section 24.8.5.5, "Continuous selection format" , added note at the end.
Analog-to-digital converter	Modified the text in Section 27.4.2, "Analog clock generator and conversion timings" . Added note to Section 27.3.2.1, "Main Configuration Register (MCR)" Added Section 27.3.8, "Power Down Exit Delay Register (PDEDL)" Deleted ABORT and ABORTCHAIN feature Added a Note in NSTART bit field of Section 27.3.2.1, "Main Configuration Register (MCR)"
Nexus Debug Interface	Added Section 38.5.1, "NPC_HNDSHK module"

NOTE

This revision history uses clickable cross-references for ease of navigation. The numbers and titles in each cross-reference are relative to the latest published release.

A.4 Changes between revisions 1 and 2

Table A-4. Changes between revisions 1 and 2

Chapter	Description
Throughout	Formatting, spelling, and grammar
Overview	Removed abort feature from the Flexcan feature list Added the Block Diagram before the Application examples In Figure 4 and Figure 5 , added blocks for RGM, PCU, CGM, ME Reversed the direction of signals from imager to device in Figure 3
Memory Map	Changed the entry MC_CGM to Clock related modules. Added a note referencing to the memory map
Signal Description	<ul style="list-style-type: none"> In the 64- and 100-LQFP pin out diagrams, revised VDD_HV and VDD_LV pins In the Supply pin table, changed the description of Ballast Voltage . Changed the column name Symbol to Multi-bonded Supplies. Added a Port Pin column Clarified the peripherals in the following port pins of the Pin Muxing table: C5, A3, A8, A10, A12, A15, C3, C4, C5, C6, and C12
Clock Architecture	<ul style="list-style-type: none"> Added CMU section (Picked the content from CMU chapter as per suggestion . Retired the CMU chapter) Copied the memory map from MC_CGM chapter Removed IRCON_STBY as suggested in shared review from IRC Oscillator Control Register Put a note below IRC_CTL register
Clock Generation Module	Changes in the memory map
Mode Entry Module	In STOP0 Mode Configuration Register, FMPLL_0ON bit change to a read-only bit
Reset Generation Module	Removed the Destructive Event Reset Disable register (RGM_DERD register) Edits in External Reset section
Interrupt Controller	In the Interrupt Vector table, edits in the Resource of 51, 52, and 54
System Status and Configuration Module	<ul style="list-style-type: none"> Removed Password Comparison registers from the memory map Removed SSCM Control Register Removed User Option Status Register Renamed Processor Start Address Register to Primary Boot Address Register in description Added Debug Status Port Signals table
System Integration Unit Lite	<ul style="list-style-type: none"> PSMI number corrected in the memory map and register description (PSMI 0 – PSMI 25) Memory map MPGPDO register address corrected PSMI 16 in Table 109 values corrected
Flash Memory	In the Platform Flash Configuration Register 1 (PFCR1), reserved 23 bit In the Platform Flash Access Protection Register (PFAPR), reserved bit 8, 9, 10 Changed description of 0 configuration of BK1_RWWC bit in PFCR0 and PFCR1 register
DMACHMUX	DMA mapping table removed ADC 1 entry Added a section Section 20.6.3, "Freezing in STOP and HALT mode"
Video Encoder Wrapper	Register formats of Control and Status registers changed Added a footnote in the Memory Map table Edits done Added a note in the Programming Sequence section

Table A-4. Changes between revisions 1 and 2 (continued)

Chapter	Description
Deserial Serial Peripheral Interface	Formatted the field description tables Added Fast Continuous Selection Added FCPCS bit in the Module Configuration Register
Analog-to-digital converter	In the feature list specified the channel and corrected the channel numbers Section 27.2, "Introduction" , channel 96 corrected to 8 Incorporated TKT064942: Added CWSEL0 and CWENR0 registers Corrected WTISR and WTIMR register descriptions
Enhanced Motor Control Timer (eTimer)	<ul style="list-style-type: none"> • Revised External Signal Description and TAI section to reflect one auxillary input support • Updated the Primary Count Source Values and Secondary Count Source Values tables to show the reserved values (auxillary inputs 0, 1, 3, 4, 5, 6, and 7 are reserved) • Added ADC Trigger section
Fault Collection Unit	<ul style="list-style-type: none"> • FCU_FFR register: Added SRF0 and SRF1 • Changed FCU_KR to Read/Write in figure • FCU_MCSR changed the description of MCAS • Changed the reset value of FCU_MCSR to 3 • Changed the description of Polarity Select bit description of FCU_MCR register Changed the bit description of FCU_FMCSR to FCU_MCSR
IEEE 1588	Added description for PTP Frame Transmission section
Nexus Debug Interface	Added value for DC and PIN fields in NDI Device ID register
Temperature Sensor	In the table TSENS calibration constants, changed all descriptions Added a footnote that defines Tsens_code Added formula for calculation the constants



How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

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

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. All other product or service names are the property of their respective owners.

© 2009 - 2015 Freescale Semiconductor, Inc.

Document Number: MPC5604ERM
Rev. 5
July 2015

