



MAX32520 Secure SOC User's Guide

Maxim Integrated, Inc.
160 Rio Robles
San Jose, CA 95134

Disclaimer**LIFE SUPPORT POLICY**

MAXIM'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF MAXIM INTEGRATED PRODUCTS, INC.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

©2021 by Maxim Integrated, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. MAXIM INTEGRATED, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. MAXIM ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering or registered trademarks of Maxim Integrated, Inc. All other product or service names are the property of their respective owners.

Arm® and Thumb® are registered trademarks of Arm Limited in the European Union and other countries. All other product or service names are the property of their respective owners.

Table of Contents

1.0	Overview.....	16
2.0	Memory Map	17
2.1	Memory Protection Unit (MPU)	17
2.2	Internal Memory Region	17
2.3	Peripheral Bus Region.....	18
2.3.1	Secure Key Storage Register	19
2.4	AHB Region.....	20
2.5	System Bus Interconnect.....	20
3.0	Digital Peripherals and System Control Registers	21
4.0	System Clocks, Reset, and Power Management	22
4.1	Clock Sources.....	22
4.1.1	120MHz Internal Clock	23
4.1.2	7.3728MHz Internal Oscillator	23
4.1.3	8kHz Ultra-Low Power Nano-Ring Internal Oscillator	23
4.1.4	Internal Clocks.....	23
4.1.5	Clock Switching.....	23
4.1.6	System Clock in Reset mode	24
4.2	Power Management.....	24
4.2.1	ACTIVE Mode	24
4.2.2	SLEEP Low-Power Mode	24
4.2.3	DEEPSLEEP Low Power Mode.....	25
4.2.4	BACKUP Low-Power Mode	26
4.2.5	SHUTDOWN Mode	27
4.2.6	Power Fail Warning.....	28
4.3	Device Resets	28
4.3.1	Peripheral Reset	28
4.3.2	Soft Reset.....	28
4.3.3	System Reset.....	28
4.3.4	Power-On Reset	28
4.4	Global Control Registers (GCR)	30
4.4.1	System Control Register (SCON, Offset 0x0000)	31
4.4.2	Reset Register 0 (RSTR0, Offset 0x0004).....	32
4.4.3	Clock Control Register (CLKCN, Offset 0x0008)	33
4.4.4	Power Management Register (PM, Offset 0x000C)	35
4.4.5	Peripheral Clock Divider Register (PCKDIV, Offset 0x0018).....	36
4.4.6	Peripheral Clock Disable Register (PERCKCN, Offset 0x0024)	37
4.4.7	Memory Clock Control Register (MEMCKCN, Offset 0x0028).....	38
4.4.8	Memory Zeroize Control Register (MEMZCN, Offset 0x002C).....	39
4.4.9	System Status Register (SYSST, Offset 0x0040)	40
4.4.10	Reset 1 Register (RSTR1, Offset 0x0044)	41
4.4.11	Peripheral Clock Disable Register 1 (PERCKCN1, Offset 0x0048).....	42
4.4.12	Event Enable Register (EVTEN, Offset 0x004C).....	43
4.4.13	Revision Register (REVISION, Offset 0x0050)	44
4.4.14	System Status Interrupt-Enabled Register (SYSSIE, Offset 0x0054).....	45
4.4.15	ECC Error Register (ECCERR, Offset 0x0064)	46
4.4.16	ECC Not Double Error Detect Register (ECCNDED, Offset 0x0068)	47
4.4.17	ECC Interrupt Enable Register (ECC_INT_EN, Offset 0x006C).....	48
4.4.18	ECC Enable Register (ECCEN, Offset 0x0000).....	49
4.5	Low-Power Management Register	50
4.5.1	Low-Power Control Register (LPCN, Offset 0x0000)	51
4.5.2	Low-Power I/O Wakeup Status Register 0 (LPWKST0, Offset 0x0004).....	52
4.5.3	Low-Power I/O Wakeup Enable Register 0 (LPWKEN0, Offset 0x0008).....	53
4.5.4	Low-Power I/O Wakeup Status Register 1 (LPWKST1, Offset 0x000C)	54
4.5.5	Low-Power I/O Wakeup Enable Register 1 (LPWKEN1, Offset 0x0010).....	55

4.5.6	Low-Power Peripheral Wakeup Status Register (LPPWST, Offset 0x0030)	56
4.5.7	Low-Power Memory Shutdown Register (LPMEMSD, Offset 0x0040)	57
4.5.8	Backup Mode General Register 0 (BUGPR0, Offset 0x0048)	58
4.5.9	Backup Mode General Register 1 (BUGPR1, Offset 0x004C)	59
5.0	System Initialization	60
5.1	System Initialization Block (SIB)	61
5.2	System Initialization Registers (SIR)	61
5.2.1	System Initialization Status Register (SISTAT, Offset 0x0000)	62
5.2.2	System Initialization Error Address Register (ERRADDR, Offset 0x0004)	63
5.2.3	Function Status Register (FNSTAT, Offset 0x0100)	64
5.2.4	Security Function Status Register (SFNSTAT, Offset 0x0104)	65
6.0	Interrupts	66
6.1	Nested Vectored Interrupt Controller	66
7.0	Instruction Cache Controller	67
7.1	Instruction Cache Controller Registers	67
7.1.1	Cache ID Register (CACHE_ID, Offset 0x0000)	68
7.1.2	Memory Configuration Register (MEMCFG, Offset 0x0004)	69
7.1.3	Cache Control and Status Register (CACHE_CTRL, Offset 0x0100)	70
7.1.4	Invalidate All (INVALIDATE, Offset 0x0700)	71
8.0	Embedded ICE and Debug	72
9.0	General-Purpose Input/Output (GPIO) Pins	73
9.1	Usage	73
9.1.1	Reset State	73
9.1.2	Input Mode Configuration	73
9.1.3	Output Mode Configuration	74
9.1.4	Alternate Function Configuration	74
9.2	Configuring GPIO (External) Interrupts	74
9.2.1	GPIO Interrupt Handling	74
9.2.2	Using GPIO for Wakeup from Low-Power Modes	75
9.3	GPIO Registers	76
9.3.1	GPIO Function Enable Bit 0 Register (GPIO_EN0, Offset 0x0000)	77
9.3.2	GPIO Function Enable Bit 0 Set Register (GPIO_EN0_SET, Offset 0x0004)	78
9.3.3	GPIO Function Enable Bit 0 Clear Register (GPIO_EN0_CLR, Offset 0x0008)	79
9.3.4	GPIO Output Enable Register (GPIO_OUT_EN, Offset 0x000C)	80
9.3.5	GPIO Output Enable Set Register (GPIO_OUT_EN_SET, Offset 0x0010)	81
9.3.6	GPIO Output Enable Clear Register (GPIO_OUT_EN_CLR, Offset 0x0014)	82
9.3.7	GPIO Output Register (GPIO_OUT, Offset 0x0018)	83
9.3.8	GPIO Output Set Register (GPIO_OUT_SET, Offset 0x001C)	84
9.3.9	GPIO Output Clear Register (GPIO_OUT_CLR, Offset 0x0020)	85
9.3.10	GPIO Input Register (GPIO_IN, Offset 0x0024)	86
9.3.11	GPIO Interrupt Mode Register (GPIO_INT_MOD, Offset 0x0028)	87
9.3.12	GPIO Interrupt Polarity Register (GPIO_INT_POL, Offset 0x002C)	88
9.3.13	GPIO Input Enable Register (GPIO_IN_EN, Offset 0x0030)	89
9.3.14	GPIO Interrupt Enable Register (GPIO_INT_EN, Offset 0x0034)	90
9.3.15	GPIO Interrupt Enable Set Register (GPIO_INT_EN_SET, Offset 0x0038)	91
9.3.16	GPIO Interrupt Enable Clear Register (GPIO_INT_EN_CLR, Offset 0x003C)	92
9.3.17	GPIO Interrupt Status Register (GPIO_INT_STAT, Offset 0x0040)	93
9.3.18	GPIO Interrupt Clear Register (GPIO_INT_CLR, Offset = 0x0048)	94
9.3.19	GPIO Wake Enable Register (GPIO_WAKE_EN, Offset 0x004C)	95
9.3.20	GPIO Wake Enable Set Register (GPIO_WAKE_EN_SET, Offset 0x0050)	96
9.3.21	GPIO Wake Enable Clear Register (GPIO_OUT_EN_CLR, Offset 0x0054)	97
9.3.22	GPIO Interrupt Dual Edge Mode Register (GPIO_INT_DUAL_EDGE, Offset 0x005C)	98
9.3.23	GPIO Input Mode Selection Register 0 (GPIO_PDPU_SEL0, Offset 0x0060)	99
9.3.24	GPIO Input Mode Selection Register 1 (GPIO_PDPU_SEL1, Offset 0x0064)	100
9.3.25	GPIO Function Enable Bit 1 Register (GPIO_EN1, Offset 0x0068)	101
9.3.26	GPIO Function Enable Bit 1 Set Register (GPIO_EN1_SET, Offset 0x006C)	102

9.3.27	GPIO Function Enable Bit 1 Clear Register (GPIO_EN1_CLR, Offset 0x0070).....	103
9.3.28	GPIO Drive Strength Register 0 (GPIO_DS_SEL0, Offset 0x00B0).....	104
9.3.29	GPIO Drive Strength Register 1 (GPIO_DS_SEL1, Offset 0x00B4).....	105
9.3.30	GPIO Pulldown/Pullup Strength Register (GPIO_PSSEL, Offset 0x00B8).....	106
10.0	Cryptographic Accelerator	107
10.1	Crypto Buffers.....	108
10.1.1	Direct Memory Access.....	108
10.1.2	FIFOs	108
10.2	Symmetric Block Cipher Accelerator.....	109
10.2.1	AES GCM and CCM	110
10.3	Hash Function Accelerator	110
10.4	Galois Field Accelerator.....	111
10.4.1	Cyclic Redundancy Check	111
10.4.2	Linear Feedback Shift Register	113
10.4.3	Entropy.....	113
10.5	Hamming Code Generator.....	113
10.6	Secure Crypto Accelerator (SCA)	116
10.6.1	Basic Operations	116
10.6.2	Manual Parameters.....	122
10.6.3	Memory Allocation for SCA Data in System Memory	125
10.7	Cryptographic Registers	130
10.7.1	Crypto Control Register (CRYPTO_CTRL, Offset 0x0000).....	131
10.7.2	Cipher Control Register (CIPHER_CTRL, Offset 0x0004).....	133
10.7.3	HASH Control Register (HASH_CTRL, Offset 0x0008).....	135
10.7.4	CRC Control Register (CRC_CTRL, Offset 0x000C).....	136
10.7.5	Crypto DMA Source Register (DMA_SRC, Offset 0x0010).....	137
10.7.6	Crypto DMA Destination Register (DMA_DEST, Offset 0x0014)	138
10.7.7	Crypto DMA Count Register (DMA_CNT, Offset 0x0018).....	139
10.7.8	Crypto Data In Register 0 (CRYPTO_DIN_0, Offset 0x0020)	140
10.7.9	Crypto Data In Register [3:1].....	140
10.7.10	Crypto Data Out Register 0 (CRYPTO_DOUT_0, Offset 0x0030).....	141
10.7.11	Crypto Data Out Register [3:1]	141
10.7.12	CRC Polynomial Register (CRC_POLY, Offset 0x0040)	142
10.7.13	CRC Value Register (CRC_VAL, Offset 0x0044)	143
10.7.14	Pseudo-Random Number Generator Register (CRC_PRNG, Offset 0x0048)	144
10.7.15	Hamming ECC Register (HAM_ECC, Offset 0x004C).....	145
10.7.16	Cipher Initial Vector Register 0 (CIPHER_INIT_0, Offset 0x0050).....	146
10.7.17	Cipher Initial Vector Register [3:1]	146
10.7.18	Cipher Key Register 0 (CIPHER_KEY_0, Offset 0x0060)	147
10.7.19	Cipher Key Register [7:1].....	147
10.7.20	HASH Message Digest Register 0 (HASH_DIGEST_0, Offset 0x0080).....	148
10.7.21	HASH Digest Register [15:1]	148
10.7.22	HASH Message Size Register 0 (HASH_MSG_SZ_0, Offset 0x00C0).....	149
10.7.23	HASH Message Size Register [3:1]	149
10.7.24	AAD Length Register 0 (A_LENGTH_0, Offset 0x00D0)	150
10.7.25	AAD_Length Register 1	150
10.7.26	PLD Length Register 0 (PLD_LENGTH_0, Offset 0x00D8).....	151
10.7.27	PLD_Length Register 1	151
10.7.28	TAG/MIC Register 0 (TAGMIC_0, Offset 0x00E0).....	152
10.7.29	TAGMIC Register [3:1].....	152
10.7.30	SCA Control Register (SCA_CN, Offset 0x0100).....	153
10.7.31	SCA Advanced Control Register (SCA_ACN, Offset 0x0104).....	155
10.7.32	SCA Status Register (SCA_ST, Offset 0x0008).....	156
10.7.33	Point P Address Register (SCA_PPi_ADDR, Offset 0x010C, 0x0110 and 0x0114).....	158
10.7.34	Point Q Address Register (SCA_PQi_ADDR, Offset 0x0118, 0x011C and 0x0120)	159
10.7.35	RDSA Address Register (SCA_RDSA_ADDR, Offset 0x0124).....	160

10.7.36	Result Address Register (SCA_RES_ADDR, Offset 0x0128)	161
10.7.37	Operation Buffer Address Register (SCA_OP_BUFF_ADDR, Offset 0x012C)	162
10.7.38	SCA Modulo Data Register (SCA_MODDATA, Offset 0x0130)	163
11.0	DMA Controller	164
11.1	DMA Channel Operation	164
11.2	DMA Channel Arbitration and DMA Bursts.....	165
11.3	DMA Source and Destination Addressing	165
11.4	Data Movement from Source to DMA FIFO.....	166
11.5	Data Movement from the DMA FIFO to Destination	167
11.6	Memory Buffer Alignment.....	168
11.7	Count-to-Zero Condition	168
11.8	Chaining Buffers	168
11.9	DMA Interrupts	169
11.10	Channel Timeouts.....	169
11.10.1	Global 24-bit Prescaler.....	169
11.10.2	8-bit Timer/Counter	170
11.11	Channel and Register Access Restrictions	170
11.12	Memory-to-Memory DMA	170
11.13	DMA Registers.....	171
11.13.1	DMA Control Register (DMA_CN, Offset 0x0000).....	172
11.13.2	DMA Interrupt Register (DMA_INT, Offset 0x0004).....	173
11.13.3	DMA Channel 0 Configuration Register (DMA0_CFG, Offset 0x0100).....	174
11.13.4	DMA Channel 0 Status Register (DMA0_ST, Offset 0x0104).....	176
11.13.5	DMA Channel 0 Source Register (DMA0_SRC, Offset 0x0108).....	177
11.13.6	DMA Channel 0 Destination Register (DMA0_DST, Offset 0x010C).....	178
11.13.7	DMA Channel 0 Count Register (DMA0_CNT, Offset 0x0110)	179
11.13.8	DMA Channel 0 Source Reload Register (DMA0_SRC_RLD, Offset 0x0114)	180
11.13.9	DMA Channel 0 Destination Reload Register (DMA0_DST_RLD, Offset 0x0118)	181
11.13.10	DMA Channel 0 Count Reload Register (DMA0_CNT_RLD, Offset 0x011C).....	182
12.0	Flash Controller.....	183
12.1	Flash Memory Operations	183
12.1.1	Clock Configuration.....	183
12.1.2	Lock Protection	184
12.1.3	Flash Write.....	184
12.1.4	Flash Page Erase	184
12.1.5	Flash Mass Erase.....	184
12.1.6	Register Access Restriction.....	185
12.1.7	Interrupt	185
12.1.8	Flash Decryption.....	185
12.1.9	Flash Encrypted Region	185
12.2	Flash Controller Registers	186
12.2.1	Flash Address Register (FLSH_ADDR, Offset 0x0000)	187
12.2.2	Flash Clock Divide Register (FLSH_CLKDIV, Offset 0x0004)	188
12.2.3	Flash Control Register (FLSH_CN, Offset 0x0008).....	189
12.2.4	Flash Interrupt Register (FLSH_INT, Offset 0x0024).....	190
12.2.5	Flash Data Register 0 (FLSH_DATA0, Offset 0x0030)	191
12.2.6	Flash Data Register 1 (FLSH_DATA1, Offset 0x0034)	192
12.2.7	Flash Data Register 2 (FLSH_DATA2, Offset 0x0038).....	193
12.2.8	Flash Data Register 3 (FLSH_DATA3, Offset 0x003C)	194
12.2.1	Access Control Register (ACNTL, Offset 0x0040)	195
13.0	Inter-Integrated Circuit (I ² C).....	196
13.1	Description	196
13.2	Protocol	196
13.2.1	Bit Transfer	196
13.2.2	Addressing	197
13.2.3	General Call Address	197

13.2.4	START Byte	198
13.2.5	Clock Synchronization	198
13.2.6	Clock Stretching.....	198
13.2.7	Timeout.....	199
13.2.8	Bus Arbitration.....	200
13.3	FIFO	200
13.4	Interrupt.....	201
13.5	Interactive Receive Mode.....	201
13.6	Error	201
13.7	DMA.....	202
13.8	Master Operation	202
13.8.1	Arbitration.....	203
13.8.2	Clock Generation	203
13.8.3	Master-Transmitter.....	203
13.8.4	Master-Receiver	204
13.9	Slave Operation	204
13.9.1	Slave-Transmitter.....	204
13.9.2	Slave-Receiver	205
13.10	I ² C Registers.....	206
13.10.1	I ² C Control Register (I2CCN, Offset 0x0000).....	207
13.10.2	I ² C Status Register (I2CST, Offset 0x0004)	209
13.10.3	I ² C Interrupt 0 Register (I2CINT0, Offset 0x0008)	211
13.10.4	I ² C Interrupt Enable 0 Register (I2CINTEN0, Offset 0x000C)	213
13.10.5	I ² C Interrupt 1 Register (I2CINT1, Offset 0x0010)	215
13.10.6	I ² C Interrupt Enable 1 Register (I2CINTEN1, Offset 0x0014).....	216
13.10.7	I ² C FIFO Configuration Register (I2CFIFO, Offset 0x0018)	217
13.10.8	I ² C Receive Configuration Register (I2CRXCFG, Offset 0x001C).....	218
13.10.9	I ² C Receive Register (I2CRX, Offset 0x0020)	219
13.10.10	I ² C Transmit Configuration Register (I2CTXCFG, Offset 0x0024)	220
13.10.11	I ² C Transmit Register (I2CTX, Offset 0x0028)	221
13.10.12	I ² C Data Register (I2CDATA, Offset 0x002C)	222
13.10.13	I ² C Master Control Register (I2CMCN, Offset 0x0030)	223
13.10.14	I ² C Clock Low Register (I2CCKL, Offset 0x0034).....	224
13.10.15	I ² C Clock High Register (I2CCKH, Offset 0x0038).....	225
13.10.16	I ² C Timeout Register (I2CTO, Offset 0x0040).....	226
13.10.17	I ² C Slave Address Register (I2CSLA, Offset 0x0044)	227
13.10.18	I ² C DMA Register (I2CDMA, Offset 0x0048)	228
13.10.19	I ² C Deglitcher Register (I2CDGLTCHR, Offset 0x0000)	229
14.0	Security Monitor	230
14.1	Destructive Reset Source.....	230
14.1.1	External Sensor Test	231
14.2	Active Die Shield	231
14.3	Temperature Sensor.....	231
14.4	Secure Voltage Monitor	231
14.5	External Sensor.....	232
14.5.1	External Sensor Warning	233
14.5.2	Power Fail Warning	233
14.6	Security Monitor Registers.....	234
14.6.1	External Sensor Control Register (EXTSCN, Offset 0x0000).....	235
14.6.2	Internal Sensor Control Register (INTSCN, Offset 0x0004)	237
14.6.3	Security Alarm Register (SECALM, Offset 0x0008)	239
14.6.4	Security Diagnostic Register (SECDIAG, Offset 0x000C)	241
14.6.5	Security Monitor Status Register (SECST, Offset 0x0034)	243
15.0	Quad Serial Peripheral Interface (QSPI)	244
15.1	Four-Wire SPI Signals.....	245
15.2	Three-Wire SPI Signal	245

15.3	Pin Configuration	246
15.3.1	Four-wire SPI Pin Configuration	247
15.3.2	Three-wire SPI Pin Configuration	247
15.3.3	Dual-Mode SPI Pin Configuration	247
15.3.4	Quad-Mode SPI Pin Configuration	247
15.4	QSPI Clock Configuration	248
15.4.1	Serial Clock	248
15.4.2	SPI Peripheral Clock	248
15.4.3	Master-Mode Serial Clock Generation	248
15.4.4	Clock Phase and Polarity Control	249
15.4.5	QSPI FIFO	250
15.4.6	QSPI Interrupt and Wakeups	251
15.5	SPI Registers	252
15.5.1	QSPI FIFO Data Register (QSPI_DATA, Offset 0x0000)	253
15.5.2	QSPI Master Control Register (QSPI_MSTR_CNTL, Offset 0x0004)	254
15.5.3	SPI Transmit Packet Size Register (QSPI_TRNMT_SIZE, Offset 0x0008)	256
15.5.4	SPI Static Configuration (QSPI_STATIC_CONFIG, Offset 0x000C)	257
15.5.5	QSPI Slave Select Timing Register (QSPI_SS_TIME, Offset 0x0010)	259
15.5.6	QSPI Master Clock Configuration Register (QSPI_CLK_CONFIG, Offset 0x0014)	260
15.5.7	QSPI DMA Register (QSPI_DMA, Offset 0x001C)	261
15.5.8	QSPI Interrupt Status Register (QSPI_INT_FL, Offset 0x0020)	263
15.5.9	QSPI Interrupt Enable Register (QSPI_INT_EN, Offset 0x0024)	265
15.5.10	QSPI Wakeup Status Register (QSPI_WAKE_FL, Offset 0x0028)	267
15.5.11	QSPI Wakeup Enable Register (QSPI_WAKE_EN, Offset 0x002C)	268
15.5.12	QSPI Status Register (QSPI_STAT, Offset 0x0030)	269
16.0	Timers	270
16.1	Basic Operation	272
16.2	Timer Operating Modes	272
16.2.1	One-Shot Mode	272
16.2.2	Continuous Mode	273
16.2.3	Counter Mode	274
16.2.4	PWM Mode	274
16.2.5	Capture Mode	275
16.2.6	Compare Mode	276
16.2.7	Gated Mode	278
16.2.8	Capture/Compare Mode	278
16.3	Timer Registers	280
16.3.1	Timer Count Register (TMR_CNT, Offset 0x0000)	281
16.3.2	Timer Compare Register (TMR_CMP, Offset 0x0004)	282
16.3.3	Timer PWM Register (TMR_PWM, Offset = 0x0008)	283
16.3.4	Timer Interrupt Register (TMR_INT, Offset 0x000C)	284
16.3.5	Timer Control Register (TMR_CN, Offset 0x0010)	285
17.0	True Random Number Generator	286
17.1	Overview	286
17.2	Reading the Random Number	286
17.3	AES Key Generation	286
17.4	Health Tests	286
17.4.1	Startup tests	286
17.4.2	Repetition count test	286
17.4.3	Adaptive proportion test	286
17.5	Interrupt Management	287
17.6	TRNG Registers	287
17.6.1	TRNG Control Register (TRNG_CN, Offset 0x0000)	288
17.6.2	TRNG Status Register (TRNG_ST, Offset 0x0004)	289
17.6.3	TRNG Data Register (TRNG_DATA, Offset 0x0008)	290
18.0	UART (Serial Port)	291

18.1	Detailed Description	291
18.1.1	Selecting the Format and Baud Rate	291
18.1.2	Transferring and Receiving Data	291
18.1.3	UART Interrupts	291
18.1.4	DMA Transfers	292
18.2	UART Registers	293
18.2.1	UART Control Register (UART_CTRL, Offset 0x0000)	294
18.2.2	UART Status Register (UART_STAT, Offset 0x0004)	295
18.2.3	UART Interrupt Enable Register (UART_INT_EN, Offset 0x0008)	296
18.2.4	UART Interrupt Status Register (UART_INT_STAT, Offset 0x000C)	297
18.2.5	UART Baud Rate Register 0 (UART_BAUD0, Offset 0x0010)	298
18.2.6	UART Baud Rate Register 1 (UART_BAUD1, Offset 0x0014)	299
18.2.7	UART Data Buffer Register (UART_DATA, Offset 0x0020)	300
18.2.8	UART DMA Register (UART_DMA, Offset 0x0030)	301
19.0	Watchdog Timer	302
19.1	Enabling and Disabling the Watchdog Timer	302
19.2	Selecting the Time Delay Period	303
19.3	Watchdog Timer Operation	303
19.4	Watchdog Timer Registers	304
19.4.1	Watchdog Timer Control Register (WDT_CTRL, Offset 0x0000)	305
19.4.2	Watchdog Timer Reset Register (WDT_RST, Offset 0x0004)	307
20.0	Serial Flash Emulation Interface	308
20.1	Operation	308
20.1.1	Pin Configuration	308
20.1.2	Slave Command/Read/Write Operations	309
20.1.3	Memory Regions	310
20.1.4	Flash MPU	310
20.2	SFE Registers	310
20.2.1	SFE Configuration Register (SF_CFG, Offset 0x0400)	311
20.2.2	SFE Discoverable Parameter System Base Register (SFDP_SBA, Offset 0x0410)	312
20.2.3	Flash's System Base Address Register (FLASH_SBA, Offset 0x0414)	313
20.2.4	Flash's System Top Address Register (FLASH_STA, Offset 0x0418)	314
20.2.5	RAM's System Base Address Register (RAM_SBA, Offset 0x041C)	315
20.2.6	RAM's System Top Address Register (RAM_STA, Offset 0x0420)	316
21.0	Secure ROM	317
21.1	Introduction	317
21.1.1	Notations	317
21.1.2	Endianness	317
21.2	Description	317
21.3	Secure Lifecycle	317
21.4	Secure Boot	318
21.5	Secure Update	318
21.6	Boot Scenarios	318
21.6.1	Secure Boot	318
21.6.2	Customer Key	318
21.6.3	Secure Scenario	318
21.7	Application Images	318
21.8	Secure ROM Services	319
21.8.1	Customer Key	320
21.9	Secure Loader	321
21.9.1	SCP Session	321
21.9.2	Packets signature	321
21.9.3	OTP Programming	322
21.9.4	Applets programming	322
21.9.5	Secure Boot	322
21.10	Basic Operations	323

21.10.1	Configuring the SoC	323
21.10.2	Set up the User OTP.....	324
21.10.3	Program the Application in Embedded Flash Memory	325
21.10.4	Load & Launch Sequence	326
21.10.5	Data Files Format.....	326
21.10.6	Tools	327

List of Tables

Table 2-1. Internal Memory Region	17
Table 2-2. Peripheral Bus Region.....	18
Table 2-3. Secure Key Storage Register (Base ADDR = 0x4000_5000)	19
Table 2-4. AHB Peripheral Region.....	20
Table 2-5. System Bus Interconnect	20
Table 4-1. Reset and Low Power Mode Effects	29
Table 4-2. Global Control Register Addresses (Base ADDR = 0x4000_0000).....	30
Table 4-3. System RAM ECC Control (Base ADDR = 0x4000_6C00).....	30
Table 4-4. SCON (Offset 0x0000).....	31
Table 4-5. RSTR0 (Offset 0x0004)	32
Table 4-6. CLKCN (Offset 0x0008).....	33
Table 4-7. PM (Offset 0x000C).....	35
Table 4-8. PCKDIV (Offset 0x0018)	36
Table 4-9. PERCKCN (Offset 0x0024).....	37
Table 4-10. MEMCKCN (Offset 0x0028).....	38
Table 4-11. MEMZCN (Offset 0x002C)	39
Table 4-12. SYSST (Offset 0x0040)	40
Table 4-13. RSTR1 (Offset 0x0044)	41
Table 4-14. PERCKCN1 (Offset 0x0048).....	42
Table 4-15. EVTEN (Offset 0x004C).....	43
Table 4-16. REVISION (Offset 0x0050).....	44
Table 4-17. SYSSIE (Offset 0x0054).....	45
Table 4-18. ECCERR (Offset 0x0064).....	46
Table 4-19. ECCNDED (Offset 0x0068)	47
Table 4-20. ECC_INT_EN (Offset 0x006C)	48
Table 4-21. ECCEN (Offset 0x0000)	49
Table 4-22. Low Power Management Addresses (Base ADDR = 0x4000_6800).....	50
Table 4-23. LPCN (Offset 0x0000).....	51
Table 4-24. LPWKST0 (Offset 0x0004)	52
Table 4-25. LPWKEN0 (Offset 0x0008).....	53
Table 4-26. LPWKST1 (Offset 0x000C).....	54
Table 4-27. LPWKEN1 (Offset 0x0010).....	55
Table 4-28. LPPWST (Offset 0x0030).....	56
Table 4-29. LPMEMSD (Offset 0x0040).....	57
Table 4-30. BUGPR0 (Offset 0x0048)	58
Table 4-31. BUGPR1 (Offset 0x004C)	59
Table 5-1. System Initialization Register Addresses (Base ADDR = 0x4000_0400).....	61
Table 5-2. SISTAT (Offset 0x0000).....	62
Table 5-3. ERRADDR (Offset 0x0004)	63
Table 5-4. FNSTAT (Offset 0x0100)	64
Table 5-5. SFNSTAT (Offset 0x0104)	65
Table 6-1. Interrupt Vectors	66
Table 7-1. Instruction Cache Controller Register Address (Base ADDR = 0x4002_A000).....	67
Table 7-2. CACHE_ID (Offset 0x0000).....	68
Table 7-3. MEMCFG (Offset 0x0004).....	69
Table 7-4. CACHE_CTRL (Offset 0x0100).....	70
Table 7-5. INVALIDATE (Offset 0x0700).....	71
Table 9-1. Input Mode Configurations	75
Table 9-2. Drive Strength Configurations	75
Table 9-3. GPIO Pin Configurations	75
Table 9-4. GPIO Register Addresses (for each port).....	76
Table 9-5. GPIO_EN0 (Offset 0x0000).....	77
Table 9-6. GPIO_EN0_SET (Offset 0x0004).....	78
Table 9-7. GPIO_EN0_CLR (Offset 0x0008).....	79

Table 9-8. GPIO_OUT_EN (Offset 0x000C)	80
Table 9-9. GPIO_OUT_EN_SET (Offset 0x0010)	81
Table 9-10. GPIO_OUT_EN_CLR (Offset 0x0014)	82
Table 9-11. GPIO_OUT (Offset 0x0018)	83
Table 9-12. GPIO_OUT_SET (Offset 0x001C)	84
Table 9-13. GPIO_OUT_CLR (Offset 0x0020)	85
Table 9-14. GPIO_IN (Offset 0x0024)	86
Table 9-15. GPIO_INT_MOD (Offset 0x0028)	87
Table 9-16. GPIO_INT_POL (Offset 0x002C)	88
Table 9-17. GPIO_IN_EN (Offset 0x0030)	89
Table 9-18. GPIO_INT_EN (Offset 0x0034)	90
Table 9-19. GPIO_INT_EN_SET (Offset 0x0038)	91
Table 9-20. GPIO_INT_EN_CLR (Offset 0x003C)	92
Table 9-21. GPIO_INT_STAT (Offset 0x0040)	93
Table 9-22. GPIO_INT_CLR (Offset 0x0048)	94
Table 9-23. GPIO_WAKE_EN (Offset 0x004C)	95
Table 9-24. GPIO_WAKE_EN_SET (Offset 0x0050)	96
Table 9-25. GPIO_WAKE_EN_CLEAR (Offset 0x0054)	97
Table 9-26. GPIO_INT_DUAL_EDGE (Offset 0x005C)	98
Table 9-27. GPIO_PDPU_SEL0 (Offset 0x0060)	99
Table 9-28. GPIO_PDPU_SEL1 (Offset 0x0064)	100
Table 9-29. GPIO_EN1 (Offset 0x0068)	101
Table 9-30. GPIO_EN1_SET (Offset 0x006C)	102
Table 9-31. GPIO_EN1_CLR (Offset 0x0070)	103
Table 9-32. GPIO_DS_SEL0 (Offset 0x00B0)	104
Table 9-33. GPIO_DS_SEL1 (Offset 0x00B4)	105
Table 9-34. GPIO_PSEL (Offset 0x00B8)	106
Table 10-1. Symmetric Block Ciphers	109
Table 10-2. Hash Functions	110
Table 10-3. Common CRC Polynomials	112
Table 10-4. SCA Opcodes for Modular Operations	118
Table 10-5. SCA Opcodes for NIST P256	119
Table 10-6. Operation Buffer mapping (up to 256-bit operands)	122
Table 10-7 - Operation Buffer mapping (up to 384-bit operands)	123
Table 10-8 - Operation Buffer mapping (up to 521-bit operands)	123
Table 10-9. Cryptographic Accelerator Register Address (Base ADDR = 0x4000_1000)	130
Table 10-10. CRYPTO_CTRL (Offset 0x0000)	131
Table 10-11. CIPHER_CTRL (Offset 0x0004)	133
Table 10-12. HASH_CTRL (Offset 0x0008)	135
Table 10-13. CRC_CTRL (Offset 0x000C)	136
Table 10-14. DMA_SRC (Offset 0x0010)	137
Table 10-15. DMA_DEST (Offset 0x0014)	138
Table 10-16. DMA_CNT (Offset 0x0018)	139
Table 10-17. CRYPTO_DIN_0 (Offset 0x0020)	140
Table 10-18. CRYPTO_DOUT_0 (Offset 0x0030)	141
Table 10-19. CRC_POLY (Offset 0x0040)	142
Table 10-20. CRC_VAL (Offset 0x0044)	143
Table 10-21. CRC_PRNG (Offset 0x0048)	144
Table 10-22. HAM_ECC (Offset 0x004C)	145
Table 10-23. CIPHER_INIT_0 (Offset 0x0050)	146
Table 10-24. CIPHER_KEY_0 (Offset 0x0060)	147
Table 10-25. HASH_DIGEST_0 (Offset 0x0080)	148
Table 10-26. HASH_MSG_SZ_0 (Offset 0x00C0)	149
Table 10-27. A_LENGTH_0 (Offset 0x00D0)	150
Table 10-28. PLD_LENGTH_0 (Offset 0x00D8)	151
Table 10-29. TAGMIC_0 (Offset 0x00E0)	152

Table 10-30. SCA_CN (Offset 0x0100).....	153
Table 10-31. SCA_ACN (Offset 0x0104).....	155
Table 10-32. SCA_ST (Offset 0x0008).....	156
Table 10-33. SCA_PPi_ADDR (Offset 0x010C, 0x0110 and 0x0114).....	158
Table 10-34. SCA_PQi_ADDR (Offset 0x0118, 0x011C and 0x0120).....	159
Table 10-35. SCA_RDSA_ADDR (Offset 0x0124).....	160
Table 10-36. SCA_RES_ADDR (Offset 0x0128).....	161
Table 10-37. SCA_OP_BUFF_ADDR (Offset 0x012C).....	162
Table 10-38. SCA_MODDATA (Offset 0x0130).....	163
Table 11-1. Source and Destination Address Definition	166
Table 11-2. Inbound Data Alignment	167
Table 11-3. Outbound Data Alignment	168
Table 11-4. DMA Register Address (Base ADDR = 0x4002_8000).....	171
Table 11-5. DMA Channel Registers Offset.....	171
Table 11-6. DMA_CN (Offset 0x0000)	172
Table 11-7. DMA_INT (Offset 0x0004)	173
Table 11-8. DMA0_CFG (Offset 0x0100).....	174
Table 11-9. DMA0_ST (Offset 0x0104)	176
Table 11-10. DMA0_SRC (Offset 0x0108)	177
Table 11-11. DMA0_DST (Offset 0x010C).....	178
Table 11-12. DMA0_CNT (Offset 0x0110).....	179
Table 11-13. DMA0_SRC_RLD (Offset 0x0114).....	180
Table 11-14. DMA0_DST_RLD (Offset 0x0118).....	181
Table 11-15. DMA0_CNT_RLD (Offset 0x011C).....	182
Table 12-1. Flash Memory Configuration	183
Table 12-2. Flash Controller Register Address (Base ADDR = 0x4002_9000).....	186
Table 12-3. FLSH_ADDR (Offset 0x0000)	187
Table 12-4. FLSH_CLKDIV (Offset 0x0004).....	188
Table 12-5. FLSH_CN (Offset 0x0008).....	189
Table 12-6. FLSH_INT (Offset 0x0024).....	190
Table 12-7. FLSH_DATA0 (Offset 0x0030).....	191
Table 12-8. FLSH_DATA1 (Offset 0x0034).....	192
Table 12-9. FLSH_DATA2 (Offset 0x0038).....	193
Table 12-10. FLSH_DATA3 (Offset 0x003C).....	194
Table 12-11. ACNTL (Offset 0x0040).....	195
Table 12-12. Definition of the First Byte	197
Table 12-13. I ² C Register Address (Base ADDR = 0x4001_D000).....	206
Table 12-14. Function Control Register Address (Base ADDR = 0x4000_0800).....	206
Table 12-15. I2CCN (Offset 0x0000).....	207
Table 12-16. I2CST (Offset 0x0004).....	209
Table 12-17. I2CINT0 (Offset 0x0008).....	211
Table 12-18. I2CINTEN0 (Offset 0x000C).....	213
Table 12-19. I2CINT1 (Offset 0x0010).....	215
Table 12-20. I2CINTEN1 (Offset 0x0014)	216
Table 12-21. I2CFIFO (Offset 0x0018).....	217
Table 12-22. I2CRXCFG (Offset 0x001C)	218
Table 12-23. I2CRX (Offset 0x0020).....	219
Table 12-24. I2CTXCFG (Offset 0x0024)	220
Table 12-25. I2CTX (Offset 0x0028).....	221
Table 12-26. I2CDATA (Offset 0x002C).....	222
Table 12-27. I2CMCN (Offset 0x0030)	223
Table 12-28. I2CCKL (Offset 0x0034)	224
Table 12-29. I2CCKH (Offset 0x0038).....	225
Table 12-30. I2CTO (Offset 0x0040).....	226
Table 12-31. I2CSLA (Offset 0x0044).....	227
Table 12-32. I2CDMA (Offset 0x0048).....	228

Table 12-33. I2CDGLTCHR (Offset 0x0000).....	229
Table 14-1. Security Monitor Register Address (Base ADDR = 0x4000_4000).....	234
Table 14-2. EXTSCN (Offset 0x0000).....	235
Table 14-3. INTSCN (Offset 0x0004).....	237
Table 14-4. SECALM (Offset 0x0008).....	239
Table 14-5. SECDIAG (Offset 0x000C).....	241
Table 14-6. SECST (Offset 0x0034).....	243
Table 15-1. Four-Wire SPI Signal.....	245
Table 15-2. Three-Wire SPI Signals.....	246
Table 15-3. SPI Mode Clock Phase and Polarity Operation.....	250
Table 15-4. SPI Register Addresses (for each instance).....	252
Table 15-5. QSPI_DATA (Offset 0x0000).....	253
Table 15-6. QSPI_MSTR_CNTL (Offset 0x0004).....	254
Table 15-7. QSPI_TRNMT_SIZE (Offset 0x0008).....	256
Table 15-8. SPI_STATIC_CONFIG (Offset 0x000C).....	257
Table 15-9. QSPI_SS_TIME (Offset 0x0010).....	259
Table 15-10. QSPI_CLK_CONFIG (Offset 0x0014).....	260
Table 15-11. QSPI_DMA (Offset 0x001C).....	261
Table 15-12. QSPI_INT_FL (Offset 0x0020).....	263
Table 15-13. QSPI_INT_EN (Offset 0x0024).....	265
Table 15-14. QSPI_WAKE_FL (Offset 0x0028).....	267
Table 15-15. QSPI_WAKE_EN (Offset 0x002C).....	268
Table 15-16. QSPI_STAT (Offset 0x0030).....	269
Table 16-1. Timer Register Address Offset (For Each Instance).....	280
Table 16-2. TMR_CNT (Offset 0x0000).....	281
Table 16-3. TMR_CMP (Offset 0x0004).....	282
Table 16-4. TMR_PWM (Offset 0x0008).....	283
Table 16-5. TMR_INT (Offset 0x000C).....	284
Table 16-6. TMR_CN (Offset 0x0010).....	285
Table 17-1. TRNG Register Address (Base ADDR = 0x4004 D000).....	287
Table 17-2. TRNG_CN (Offset 0x0000).....	288
Table 17-3. TRNG_ST (Offset 0x0004).....	289
Table 17-4. TRNG_DATA (Offset 0x0008).....	290
Table 18-1. UART Register Addresses (0x40020000).....	293
Table 18-2. UART_CTRL (Offset 0x0000).....	294
Table 18-3. UART_STAT (Offset 0x0004).....	295
Table 18-4. UART_INT_EN (Offset 0x0008).....	296
Table 18-5. UART_INT_STAT (Offset 0x000C).....	297
Table 18-6. UART_BAUD0 (Offset 0x0010).....	298
Table 18-7. UART_BAUD1 (Offset 0x0014).....	299
Table 18-8. UART_DATA (Offset 0x0020).....	300
Table 18-9. UART_DMA (Offset 0x0030).....	301
Table 19-1. Watchdog Timer Interrupt Period Example.....	303
Table 19-2. Watchdog Timer Register Addresses.....	304
Table 19-3. WDT_CTRL (Offset 0x0000).....	305
Table 19-4. WDT_RST (Offset 0x0004).....	307
Table 20-1. SFE Commands and Opcodes.....	309
Table 20-2. SFE Registers Address (Base ADDR= 0x400A 0000).....	310
Table 21-1. OTP Layout.....	325

List of Figures

Figure 1-1. Block Diagram	16
Figure 4-1. Clock Sources (MAX32520)	22
Figure 4-2. SLEEP Mode Clock Control	25
Figure 4-3. DEEPSLEEP Clock Control	26
Figure 4-4. BACKUP Clock Control	27
Figure 5-1. System Initialization Block Start-up Flow	60
Figure 10-1. Crypto Accelerator Block Diagram	107
Figure 10-2. DMA Block Diagram	108
Figure 10-3. Galois Field CRC and LFSR Architecture	113
Figure 10-4. Hamming XOR Calculation	114
Figure 11-1. DMA Controller Block Diagram	164
Figure 12-1. I ² C Start and Stop Conditions	197
Figure 12-2. General Call Address, Bytes Format	197
Figure 12-3. Clock Synchronization	198
Figure 12-4. Master Arbitration	200
Figure 14-1. Security Monitor Block Diagram	230
Figure 14-2. External Sensors Block Diagram	233
Figure 15-1. QSPI Block Diagram	244
Figure 15-2. 4-Wire SPI Connection Diagram	245
Figure 15-3. Generic 3-Wire SPI Master to Slave Connection	246
Figure 15-4. Dual Mode SPI Connection Diagram	247
Figure 15-5. SCK Clock Rate Control	249
Figure 15-6. SPI Clock Polarity	249
Figure 16-1. 32 bit Timer Block Diagram	271
Figure 19-1. Watchdog Timer Block Diagram	302
Figure 21-1. Application Image Format	319
Figure 21-2. SCP Session	321
Figure 21-3. Application Programming	322
Figure 21-4. SoC Secure Application Execution	323

1.0 Overview

The MAX32520 provides an interoperable, secure, and cost-effective solution to build new generations of trusted devices for various applications including wireless access points, network devices, and IoT Gateways. This part incorporates a 32-bit Arm® Cortex®-M4F microcontroller, 2MB of encrypted flash, 136KB of system RAM+34KB ECC, 8KB of User One-Time-Programmable (OTP) memory, and 128KB of Boot ROM.

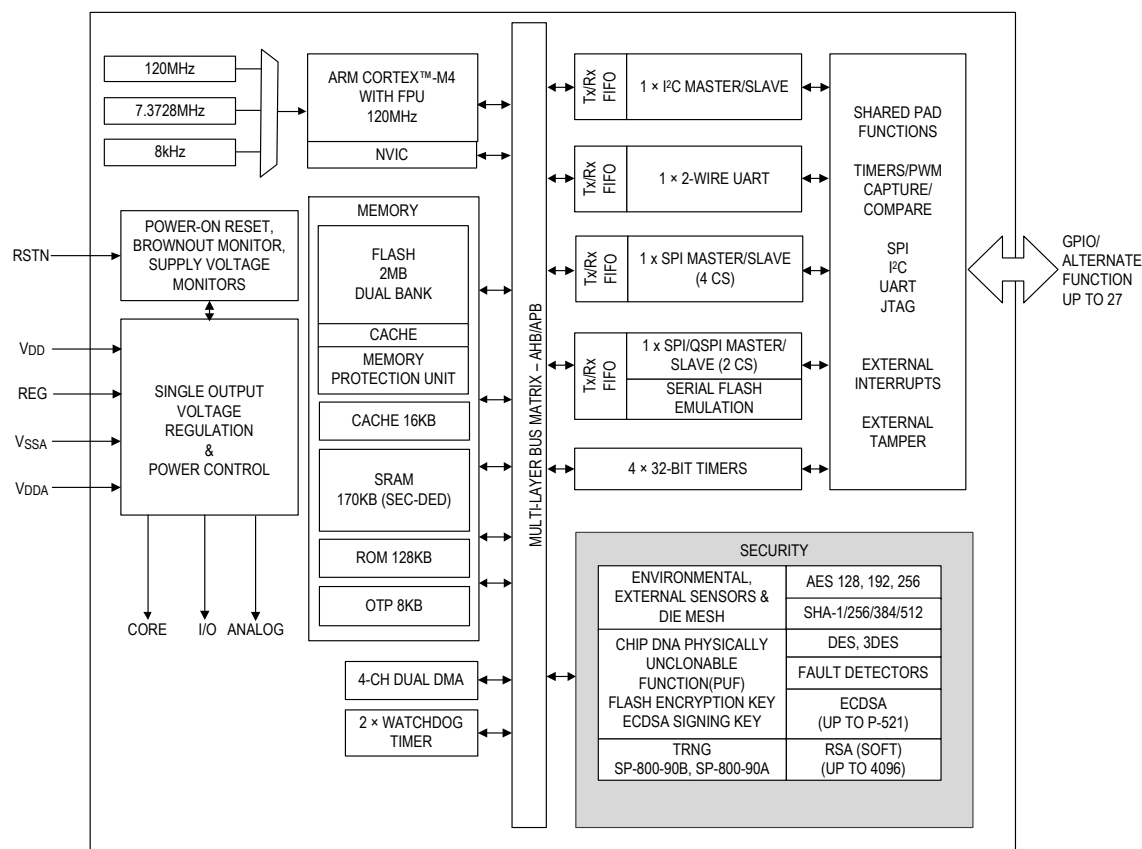
The MAX32520 incorporates Maxim's patented ChipDNA™ PUF technology. The ChipDNA technology involves a physically unclonable function (PUF) that enables cost-effective protection against invasive physical attacks. Using the random variation of semiconductor device characteristics that naturally occur during wafer fabrication, the ChipDNA circuit generates a unique output value that is repeatable over time, temperature, and operating voltage. Attempts to probe or observe ChipDNA operation modifies the underlying circuit characteristics, preventing discovery of the unique value used by the chip cryptographic functions. The MAX32520 utilizes the ChipDNA output as key content to cryptographically secure all device stored data and optionally, under user control, as the private key for the ECDSA signing operation.

In addition to hardware crypto functions, the MAX32520 provides a FIPS/NIST compliant true random number generator, environmental and tamper detection circuitry to facilitate system-level security for the application.

These microcontrollers include multiple communication interfaces. Two SPI ports, one UART, and one I²C bus are provided. There are four on-chip timers which support PWM output generation for direct control of external devices. For complete list of differences including the pinouts and ordering part numbers, refer to each device datasheet.

The SPI0 port has a serial flash emulation mode allowing direct code and data fetching from internal flash by another processor, therefore enabling a secure boot for a host microcontroller.

Figure 1-1. Block Diagram



2.0 Memory Map

The microcontroller has several memory regions. The memory map is divided into the internal memory, external memory, and peripheral space. The device has 136KB of SRAM plus 34KB ECC.

2.1 Memory Protection Unit (MPU)

The MPU is a transparent engine that complies with NIST standards to secure flash read operations as well as execution.

When the system requests an AHB Read to the flash controller, the transfer goes through this unit. The data is decrypted on-the-fly during the read operation and execution. The ROM encrypts the code prior to writing it to the flash memory through the internal cryptographic engine. The write to the flash is *not* encrypted on the fly. The MPU only handles the decryption on the fly.

2.2 Internal Memory Region

The device's internal memory region contains the program and data memories for the CPU and any peripheral functions that require bus mapped memory. The allocation for this region is shown in Table 2-1.

Table 2-1. Internal Memory Region

PROGRAM	KByte	Start Addr	End Addr
ROM	128	0000 0000	0001 FFFF
Reserved	262016	0002 0000	0FFF FFFF
Flash Bank0	1024	1000 0000	100F FFFF
Flash Bank1	1024	10100000	101F FFFF
Reserved	6144	1020 0000	107F FFFF
Maxim OTP	24	1080 0000	1080 5FFF
User OTP	8	1080 6000	1080 7FFF
Reserved	253920	1080 8000	1FFF FFFF

DATA	KByte	Start Addr	End Addr
Data SRAM	136 (+ECC)	2000 0000	2002 A7FF
Reserved	32696	2002 A800	21FF FFFF
Bit-Band Alias	32768	2200 0000	23FF FFFF
Reserved	458752	2400 0000	3FFF FFFF

2.3 Peripheral Bus Region

Table 2-2 shows the region that contains the peripheral control registers for the device. Note that the addresses shown specify the base address for each peripheral function(s). The base address represents the starting address for each peripheral (for example, the device watchdog timer 0 control registers start at address 0x4000_3000). In general, the number of peripherals allocated to a section is smaller than the size of the section. Any unused address space is to be considered reserved. The device supports the full APB 3.0 specification. For each peripheral's registers and definitions, refer to the peripheral's designated chapter.

Table 2-2. Peripheral Bus Region

APB REGION	KByte	Start Address	End Address
Global Control Registers	1	4000 0000	4000 03FF
System Initialization Registers	1	4000 0400	4000 07FF
System Function Control	1	4000 0800	4000 0BFF
Reserved	1	4000 0C00	4000 0FFF
Crypto	4	4000 1000	4000 1FFF
Reserved	4	4000 2000	4000 2FFF
Watch Dog Timer 0	1	4000 3000	4000 33FF
Watch Dog Timer 1	1	4000 3400	4000 37FF
Reserved	2	4000 3800	4000 3FFF
Security Monitor	4	4000 4000	4000 4FFF
Secure Key Storage Register	1	4000 5000	4000 53FF
Reserved	5	4000 5400	4000 67FF
Power Management	1	4000 6800	4000 6BFF
System RAM ECC Control	1	4000 6C00	4000 6FFF
Reserved	1	4000 7000	4000 73FF
Reserved	3	4000 7400	4000 7FFF
GPIO 0	4	4000 8000	4000 8FFF
GPIO 1	4	4000 9000	4000 9FFF
Reserved	24	4000 A000	4000 FFFF
Timer 0	4	4001 0000	4001 0FFF
Timer 1	4	4001 1000	4001 1FFF
Timer 2	4	4001 2000	4001 2FFF
Timer 3	4	4001 3000	4001 3FFF
Reserved	36	4001 4000	4001 CFFF
I2C	4	4001 D000	4001 DFFF
Reserved	8	4001 E000	4001 FFFF

APB REGION	KByte	Start Address	End Address
UART	4	4002 0000	4002 0FFF
Reserved	28	4002 1000	4002 7FFF
DMA	4	4002 8000	4002 8FFF
Flash Controller	4	4002 9000	4002 9FFF
Cache Controller	4	4002 A000	4002 AFFF
Reserved	92	4002 B000	4004 1FFF
Reserved	4	4004 2000	4004 2FFF
Reserved	12	4004 3000	4004 5FFF
SPI 0	4	4004 6000	4004 6FFF
SPI 1	4	4003 7000	4003 7FFF
Reserved	4	4004 8000	4004 CFFF
TRNG	4	4004 D000	4004 DFFF
Reserved	28	4004 E000	4009 FFFF
Serial Flash Emulation (SFE)	4	400A 0000	400A 0FFF
Reserved	60	400A 1000	400A FFFF

2.3.1 Secure Key Storage Register

Table 2-3 summarizes the content and offset addresses of the keys and their usage.

Table 2-3. Secure Key Storage Register (Base ADDR = 0x4000 5000)

Key Name	Offset	Description
AES_KEY0[127:0]	0x000	128-bit AES Key 0 Register
AES_KEY1[127:0]	0x010	128-bit AES Key 1 Register. This key is used in conjunction with Key 0 to form a 256-bit key for 256-AES mode.
PUF_KEY1[255:0]	0x020	256-bit PUF Key 1 Register. At power-up, this key is loaded with a 256-bit PUF key. This key is used by MPU.
PUF_KEY2[255:0]	0x040	256-bit PUF Key 2 Register. At power-up, this key is loaded with a 256-bit PUF key. This key can be selected by setting SCA_CN.HWKEY for ECDSA operations.

2.4 AHB Region

Table 2-4 shows the AHB region along with the base addresses. No peripherals are mapped onto the AHB.

Table 2-4. AHB Peripheral Region

AHB REGION	KByte	Start Addr	End Addr
Reserved	329	400B 0000	400F FFFF

2.5 System Bus Interconnect

The system bus interconnect is summarized in Table 2-5. The bus arbitration scheme is fixed burst.

Table 2-5. System Bus Interconnect

	DMA	System Controller	Crypto DMA	Serial Flash Emulation
System RAM0	X	X	X	X
System RAM1	X	X	X	X
System RAM2	X	X	X	X
System RAM3	X	X	X	X
System RAM4	X	X	X	X
APB Bridge	X	X		

3.0 Digital Peripherals and System Control Registers

All registers used to control the core system functions and peripheral modules are memory mapped into the AMBA Peripheral Bus (APB) memory region. This device is compliant with version 3.0 of the APB specification as well as the Secure Arm Platform Specification.

Reserved register bits are designated by “RFU”. Reserved bits should only be written as 0 unless otherwise stated.

The reset value is normally specified. However, some registers may have different reset values depending on the reset conditions. The following defines other possible reset values:

- N/A – Not affected by this reset condition. This normally applies to registers with limited reset sources.
- S – Not specified. This applies to registers with no specify initial value or uninitialized.

The access field of each register bit is defined as follows:

- R – Read allowed. Additional restriction may apply. See register definition for details.
- W – Write allowed. Additional restriction may apply. See register definition for details.
- S – Set to 1 only. Clear to 0 ignored.
- C – Clear to 0 only. Writes to 1 ignored.
- W1C – Write 1 to Clear.

4.0 System Clocks, Reset, and Power Management

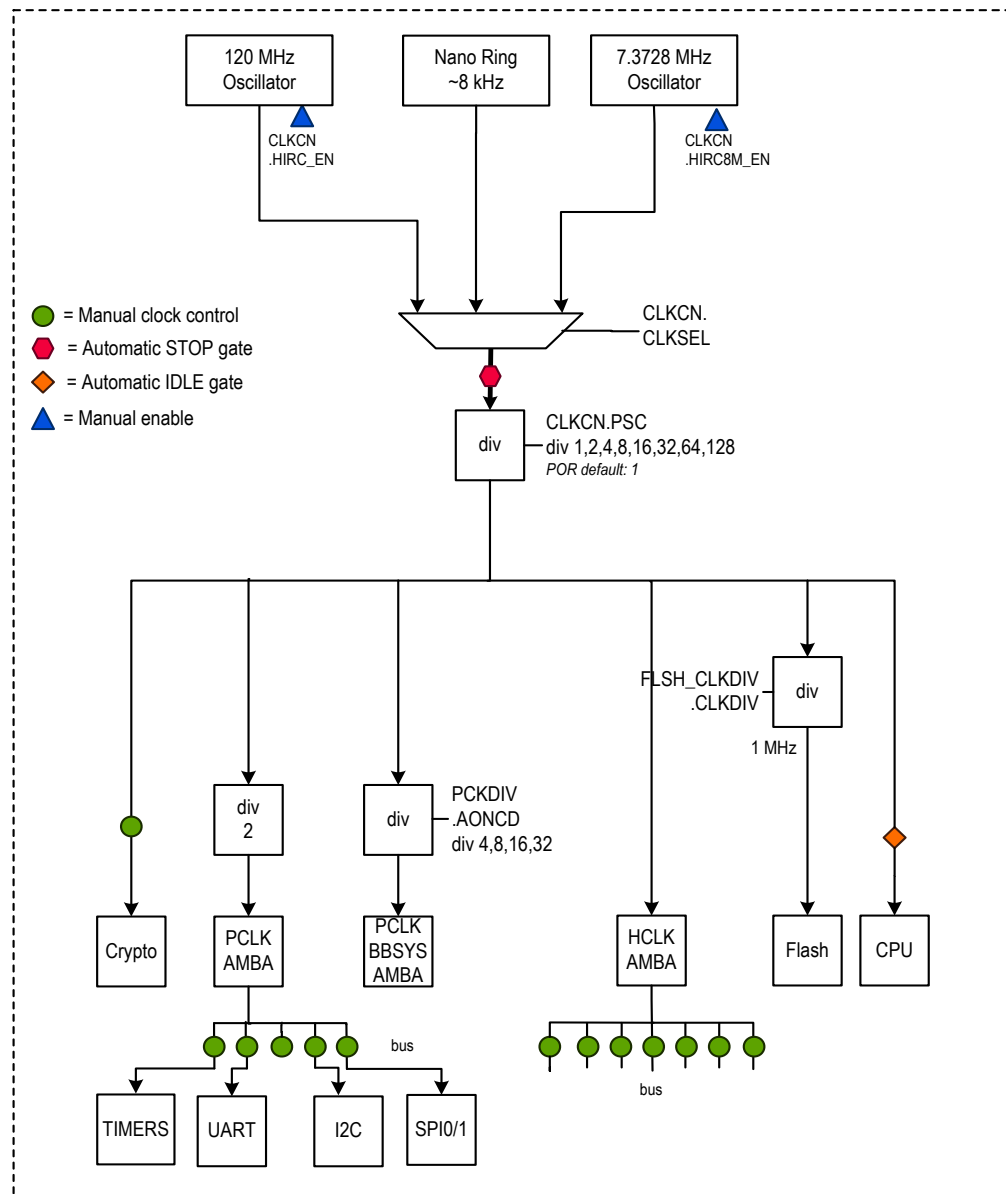
4.1 Clock Sources

The internal clock circuitry generates the system clock from one of the following internal clock sources:

- High accuracy oscillator at 120MHz
- High accuracy oscillator at 7.3728MHz
- Nano Ring oscillator at 8kHz

A simplified clock distribution block diagrams is shown in Figure 4-1.

Figure 4-1. Clock Sources (MAX32520)



4.1.1 120MHz Internal Clock

This is the fastest oscillator and draws the most power. This oscillator is automatically enabled after a Power-On Reset (POR) and following a System Reset. The high-speed oscillator is disabled when the device is in DEEPSLEEP and BACKUP mode.

4.1.2 7.3728MHz Internal Oscillator

This is a very low-power internal oscillator that can be selected as the System Clock. Firmware selection of the voltage that controls this oscillator is controlled by the register bit CLKCN.HIRC8M_VS.

The internal CPU core supply voltage (V_{CORE}) is the default option. The external pin V_{DDA} can also be selected. The V_{DDA} pin goes to an internal 1V regulator that also provides the analog supply voltage for this device.

This oscillator is disabled when the device is in DEEPSLEEP and BACKUP mode.

This oscillator is enabled by default at power-up.

4.1.3 8kHz Ultra-Low Power Nano-Ring Internal Oscillator

This is an ultra-low power internal oscillator that can be selected as the System Clock. This oscillator is enabled at power-up and cannot be disabled by the firmware.

4.1.4 Internal Clocks

Internal clocks are generated directly from the System Clock. The System Clock is sourced from one of three clock sources: 120MHz, 7.3728MHz, or 8kHz oscillators.

Additionally, the CPU clock is taken from the output of a prescaler that can divide the source clock by 1, 2, 4, 8, 16, or 32 as follows:

PSC[2:0]	CPU Clock
000	/1 (120MHz)
001	/2 (60MHz)
010	/4 (30MHz)
011	/8 (15MHz)
100	/16 (7.5MHz)
101	/32 (3.75MHz)
110	/64 (1.875MHz)
111	/128 (0.9375MHz)

The prescaler is controlled via the CLKCN register. See Section 4.4.3 on page 33. The output of the prescaler provides the AHB bus clock. The AHB bus clock is then divided by 2 to generate the APB bus clock.

4.1.5 Clock Switching

Before using any oscillator, the desired oscillator must first be enabled by setting the oscillator's enable bit in the CLKCN register. Once an oscillator's enable bit is set, the oscillator's ready bit must read 1 prior to attempting to use the oscillator as a system oscillator source. The oscillator ready status flags are contained in the CLKCN register.

Once the corresponding oscillator ready bit is set, the oscillator can then be selected as the System Clock by configuring the Clock Source Select field (CLKCN.CLKSEL).

Any time the firmware changes the System Clock by changing CLKCN.CLKSEL, the Clock Ready bit CLKCN.CKRDY is automatically cleared to indicate that a System Clock switchover is in progress. When switchover is complete, CLKCN.CKRDY is set to 1 by the hardware, indicating the oscillator selected is ready for use.

4.1.6 System Clock in Reset mode

On Power-On Reset (POR) and System Reset, all oscillator states are reset to the default: The 120MHz and 8kHz oscillators are enabled, while the 7.3728MHz oscillator is disabled. The oscillators are not reset on Soft Reset or Peripheral Reset.

4.2 Power Management

This chip supports five power management modes. ACTIVE is the highest performance operating mode. Any low power state can wake up to ACTIVE by a wakeup event. Wakeup events include any external or internal interrupt, and Watchdog Interrupt.

The Arm Cortex-M family of CPUs have two built-in low-power modes, designated SLEEP and DEEPSLEEP. The implementation of these low-power modes are specific to the microcontroller's design. These modes are enabled using the System Control Register (SCR), an Arm Cortex System Control Block register. Write register bit SCR.sleepdeep to select the low power mode as shown in the pseudocode below.

```
SCR.sleepdeep = 0; // SLEEP mode enabled  
SCR.sleepdeep = 1; // DEEPSLEEP mode enabled
```

Once enabled, the device enters the enabled low-power mode when either a WFI (Wait For Interrupt) or WFE (Wait For Event) instruction is executed.

Immediately before entering any low-power mode, enable the System Clock to be used in that low-power mode. If DEEPSLEEP or BACKUP is to be entered, ensure the selected System Clock is not automatically disabled in that low power mode. If the selected System Clock is disabled in that low power mode, it will be enabled upon returning to ACTIVE mode.

Refer to the Arm Cortex-M4 core reference manual for more information on SCR.

4.2.1 ACTIVE Mode

This is the highest performance mode. All internal clocks, registers, memory, and peripherals are enabled. The CPU is running and executing application code. All oscillators are available.

Dynamic clocking allows the firmware to selectively enable or disable clocks and power to individual peripherals, providing the optimal mix of high-performance and power conservation. The internal RAMs that can be enabled, disabled, or placed in low-power RAM Retention Mode include data SRAM memory blocks, on-chip caches, and on-chip FIFOs.

4.2.2 SLEEP Low-Power Mode

This is a low-power mode that suspends the CPU with a fast wakeup time to the ACTIVE mode. It is like the ACTIVE mode except the CPU clock is disabled, which temporarily prevents the CPU from executing code. All oscillators remain active if enabled, and the Always On Domain (AOD) and RAM retention are retained.

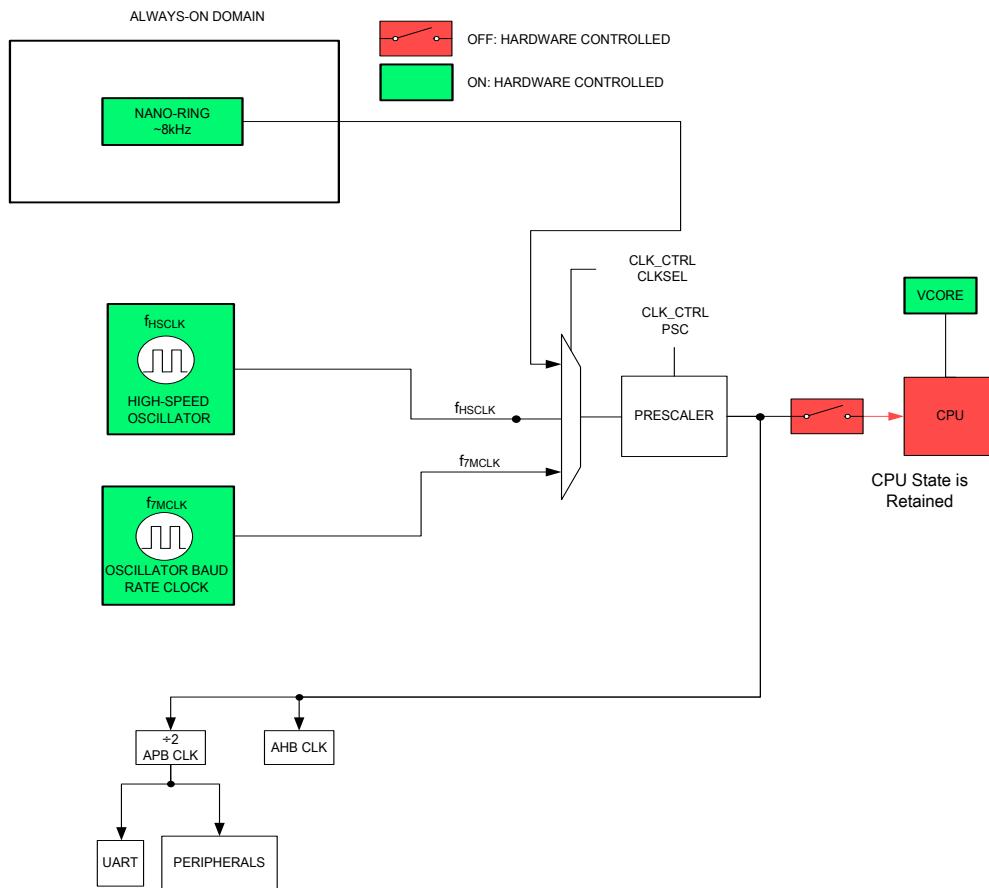
The device returns to the ACTIVE mode from any internal or external interrupt.

The following pseudocode places the device in SLEEP mode:

```
SCR.sleepdeep = 0; // SLEEP mode enabled  
WFI (or WFE); // Enter the low power mode enabled by SCR.sleepdeep
```

Figure 4-2, shows the clocks available and blocks disabled during SLEEP mode.

Figure 4-2. SLEEP Mode Clock Control



4.2.3 DEEPSLEEP Low Power Mode

This mode is like the SLEEP mode except that all internal clocks are gated off/ Also, the System Clock is gated off and the two bus clocks PCLK and HCLK are inactive. The CPU state is retained.

Because the main bus clocks are disabled, all peripherals are inactive. Only the external interrupt can return the device to ACTIVE mode.

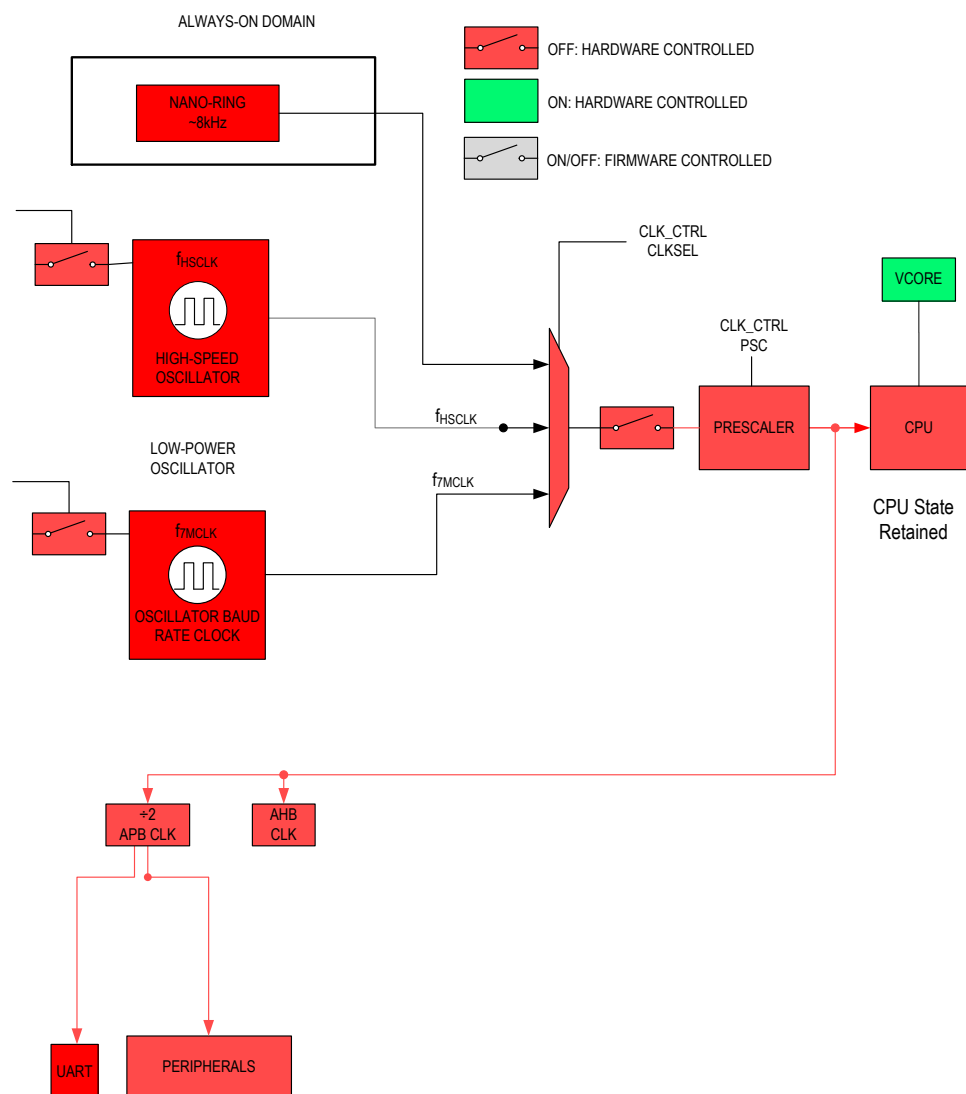
All internal register contents and all RAM contents are preserved. The GPIO pins retain their state in this mode. The Always-on Domain (AOD) and RAM Retention are available.

The following pseudocode places the device in the DEEPSLEEP mode:

```
SCR.sleepdeep = 1; // DEEPSLEEP mode enabled
WFI (or WFE);    // Enter DEEPSLEEP mode
```

The 8kHz are available. Figure 4-3, shows the clock control during the DEEPSLEEP mode.

Figure 4-3. DEEPSLEEP Clock Control



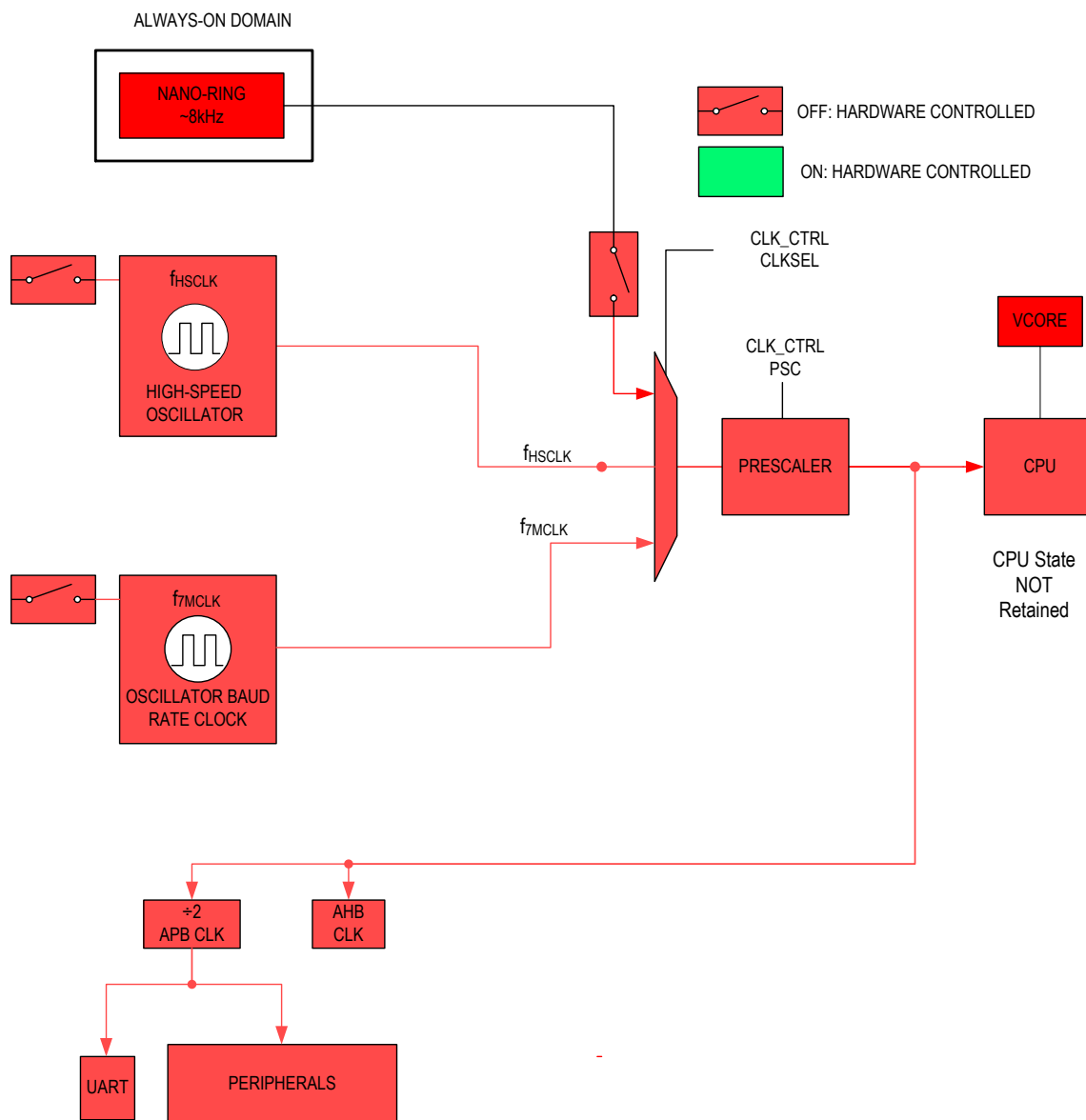
4.2.4 BACKUP Low-Power Mode

This is the lowest power operating mode. All oscillators are disabled and the CPU state is not maintained.

All the keys are retained. The RAM Retention can be configured prior to entering the BACKUP Low-Power Mode to retain the SRAM0, 1, and 4 content.

The BACKUP mode supports the same wakeup sources as the DEEPSLEEP mode. To immediately enter BACKUP mode, write `PM.MODE = 0b100`. Figure 4-4 shows the clock control during the BACKUP mode.

Figure 4-4. BACKUP Clock Control



4.2.5 SHUTDOWN Mode

The Shutdown Mode is not a low-power mode. It is intended to wipe all volatile memory from the device. In the Shutdown state, internal logic gates off all internal power. There is no data, register, or RAM retention in this mode. All wakeup sources, wakeup logic, and interrupts are disabled. The Always-on Domain (AoD) is disabled, clearing all keys in the Secure Key Storage Register. The device only recovers through a Power-On Reset (POR), which re-initializes the device.

In security-related applications it might be necessary to completely disable the device if a breach or other security threat is detected. In this situation, clear all memory including the AoD.

To immediately put the device into Shutdown, write **PM.MODE = 0b111**.

4.2.6 Power Fail Warning

In addition to Power Monitor Reset Thresholds, high and low Power Fail Warning Thresholds are also available. Interrupt #16 fires, if enabled, when the V_{DD} drops below the threshold specified in INTSCN.VTM_LOTHSEL (see Table 14-3) or V_{DD} goes above typical 3.8V threshold. Also, enabling the ICE interface triggers this interrupt.

The user software can read the status of this bit indirectly through the Security Alarm Register (SECALM).

4.3 Device Resets

Four device resets are available – Peripheral, Soft, System, and Power-On. On completion of any of the four reset cycles, all peripherals are reset. On completion of any reset, cycles HCLK and PCLK are operational, the CPU core receives clocks and power, and the device is in ACTIVE mode. Program execution begins at the reset vector address.

Contents of the Always-On Domain (AoD) do not change after any type of reset, except a POR.

Each of the on-chip peripherals can also be reset to their POR default state using the two reset registers RST0 and RST1.

4.3.1 Peripheral Reset

This resets all peripherals. The CPU retains its state. The GPIO, Watchdog Timers, AoD, RAM Retention, and General Control Registers (GCR), including the clock configuration, are unaffected.

To start a Peripheral Reset, set RST0.PRST = 1.

4.3.2 Soft Reset

This is the same as a Peripheral Reset except that it also resets the GPIO to its Power-On Reset state. All I/O functions input are tri-stated.

To start a Soft Reset, set RST0.SRST = 1.

4.3.3 System Reset

This is the same as Soft Reset except it also resets all GCR, resetting the clocks to their default state. The CPU state is reset as well as the watchdog timers. The AoD and RAM Retention are unaffected.

A watchdog timer reset event initiates a System Reset. To start a System Reset from firmware, set RST0.SYSTEM = 1.

4.3.4 Power-On Reset

A POR resets everything in the device to its default state, as if power had been cycled to the device. The RAMs are cleared Table 4-1, shows the effects of the reset types and the power modes supported by this device.

Table 4-1. Reset and Low-Power Mode Effects

	Peripheral Reset	Soft Reset	System Reset	POR	ACTIVE Mode	SLEEP Mode	DEEP-SLEEP Mode	BACKUP Mode
GCR Reset	No	No	Reset	Reset	N/A	N/A	N/A	N/A
8kHz Osc	On	On	On	On	On	On	Off	Off
7.3728 MHz Osc	-	-	On	On	Y	Y	Auto Off	Off
120MHz Osc	-	-	On	On	Y	Y	Auto Off	Off
PCLK	On	On	On	On	On	On	Off	Off
HCLK	On	On	On	On	On	On	Off	Off
CPU Clock	On	On	On	On	On	Off	Off	Off
VCore	On	On	On	On	On	On	On	Off
CPU State Retention	On	On	Reset	Reset	N/A	On	On	Off
DMA	Reset	Reset	Reset	Reset	Y	Y	Off	Off
WDT1 & WDT2	-	-	Reset	Reset	Y	Y	Off	Off
GPIO	-	Reset	Reset	Reset	Y	Y	Y	Y
Flash All Cache Instances	Reset	Reset	Reset	Reset	Y	Y	Off	Off
Other Peripherals	Reset	Reset	Reset	Reset	Y	Y	Off	Off
External Reset Wakeup	-	-	-	-	-	Y	Y	Y
GPIO Wakeup	-	-	-	-	-	Y	Y	Y
AOD	On	Y	Y	Y	Y	On	On	On
RAM Retention	Y	Y	Y	Reset	Y	Y	Y	Auto Off

Table key:

Y = Enabled, can be disabled by firmware

On = Enabled by hardware (cannot be disabled)

Off = Disabled by hardware (cannot be enabled)

Auto Off = Can either be left on, or automatically gated off when in this power mode.

- = No Effect

N/A = Not Applicable

Note1: SLEEP, DEEPSLEEP, and BACKUP low-power modes wake-up directly to ACTIVE with no reset.

Note2: The Always on Domain (AoD) includes the oscillator, Secure Key Storage Register, RAM retention and sleep registers, and Low Power Wakeup Control Registers.

Note3: RAM Retention applies to data SRAM, all caches, and all FIFOs.

Note4: Peripheral, Soft, and System Resets are initiated by the firmware through the RSTO register.

Note5: A Watchdog Reset initiates a System Reset.

4.4 Global Control Registers (GCR)

The Global Control Register (GCR) set of registers controls the global functions of the MAX32520.

Address assignments for the Global Control registers are outlined in Table 4-2. Reserved register bits should only be written as 0.

Table 4-2. Global Control Register Addresses (Base ADDR = 0x4000_0000)

Offset	Access	Register	Description
0x0000	RW	SCON	System Control Register
0x0004	RW	RSTR0	Reset 0 Register
0x0008	RW	CLKCN	Clock Control Register
0x000C	RW	PM	Power Management Register
0x0010	R	RFU	Reserved for future use
0x0014	R	RFU	Reserved for future use
0x0018	RW	PCKDIV	Peripheral Clock Divider Register
0x0024	RW	PERCKCN	Peripheral Clock Disable Register
0x0028	RW	MEMCKCN	Memory Clock Control Register
0x002C	RW	MEMZCN	Memory Zeroize Control Register
0x0030	R	RFU	Reserved for future use
0x0034	R	RFU	Reserved for future use
0x0038	R	RFU	Reserved for future use
0x003C	R	RFU	Reserved for future use
0x0040	R	SYSST	System Status Register
0x0044	RW	RSTR1	Reset 1 Register
0x0048	RW	PERCKCN1	Peripheral Clock Disable Register 1
0x004C	RW	EVTEN	Event Enable Control Register
0x0050	R	REVISION	Revision Register
0x0054	RW	SYSSIE	System Status Interrupt Register
0x0058	R	RFU	Reserved for future use
0x005C	R	RFU	Reserved for future use
0x0060	R	RFU	Reserved for future use
0x0064	RW	ECCERR	ECC Error Register
0x0068	RW	ECCNDED	ECC Not Double Error Detect Register
0x006C	RW	ECCIRQEN	ECC IRQ Enable Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on reset values and various access modes.

Table 4-3. System RAM ECC Control (Base ADDR = 0x4000_6C00)

Offset	Access	Register	Description
0x0000	RW	ECCEN	ECC Enable Register (system RAM)

4.4.1 System Control Register (SCON, Offset 0x0000)

Table 4-4. SCON (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[1:0]		RFU	RFU[2:0]			OVR[1:0]	
Reset	00		*	100			00	
Access	R		R	R			RW	
Position	15	14	13	12	11	10	9	8
Field	CHKRES	RFU	CCHK	RFU[4:0]				
Reset	0	0	0	0 0000				
Access	R	R	RS	R				
Position	7	6	5	4	3	2	1	0
Field	RFU	CCACHE_FLUSH	RFU	RFU	RFU	RFU	RFU	BSTAPEN
Reset	0	0	0	0	0	0	1	*
Access	R	RW	R	R	R	R	R	*

Name	Bits	Description	Settings									
BSTAPEN	0	Boundary Scan TAP enable. When enabled, the JTAG port is connected to the Boundary Scan TAP. Otherwise, the port is connected to the Arm ICE function. This bit is reset by the POR. Reset value and access depend on the part number: <table><tr><th>Part</th><th>Reset Value</th><th>Access</th></tr><tr><td>MAX32520-LNS+</td><td>1</td><td>RW</td></tr><tr><td>MAX32520-LNJ+</td><td>0</td><td>R</td></tr></table>	Part	Reset Value	Access	MAX32520-LNS+	1	RW	MAX32520-LNJ+	0	R	0: Boundary Scan TAP port disabled. 1: Boundary Scan TAP port enabled.
Part	Reset Value	Access										
MAX32520-LNS+	1	RW										
MAX32520-LNJ+	0	R										
RFU	5:1	Reserved	N/A									
CCACHE_FLUSH	6	Code Cache Flush. This bit is used to flush the code caches and the instruction buffer of the Cortex-M4F. *Once set, the flush operation is carried to completion. This bit is cleared by the hardware when the operation is done. Clearing the bit by software does not halt the operation.	0: Normal Cache operation 1: Code Caches and CPU instruction buffer are flushed									
RFU	12:7	Reserved	N/A									
CCHK	13	Compute ROM Checksum. This bit is self-cleared when calculation is completed. Once set, software clearing this bit is ignored and the bit remains set until the operation is completed.	0: No operation. 1: Starts ROM checksum calculation.									
RFU	14	Reserved	N/A									
CHKRES	15	ROM Checksum Result. This bit is only valid when CHKRd=1.	0: Pass. 1: Fail.									
OVR	17:16	Operating Voltage Range. Setting these bits according to the V _{CORE} voltage allows the on-chip random-Access memories to operate in their optimal timing range.	00: 0.9V +/- 10% 01: 1.0V +/- 10% 10: 1.1V +/- 10% 11: Reserved									
RFU	31:18	Reserved	N/A									

4.4.2 Reset Register 0 (RSTR0, Offset 0x0004)

Table 4-5. RSTR0 (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	SYSTEM	PRST	SRST	RFU[4:0]				
Reset	0	0	0	0 0000				
Access	RS	RS	RS	R				

Position	23	22	21	20	19	18	17	16
Field	RFU[4:0]					CRYPTO	RFU	I2C0
Reset	0000 0					0	0	0
Access	R					RS	R	RS

Position	15	14	13	12	11	10	9	8
Field	RFU	SPI1	SPI0	RFU	UART0	RFU	RFU	TIMER3
Reset	0	0	0	0	0	0	0	0
Access	R	RS	RS	R	RS	R	R	RS

Position	7	6	5	4	3	2	1	0
Field	TIMER2	TIMER1	TIMER0	RFU	GPIO1	GPIO0	WDT	DMA
Reset	0	0	0	0	0	0	0	0
Access	RS	RS	RS	R	RS	RS	RS	RS

Setting a bit to 1 in this register resets the corresponding peripheral or set of peripherals. Once the reset has completed, the bit is automatically cleared to 0 by the hardware. The software writes of 0 to bits in this register have no effect.

Name	Bits	Description
DMA	0	DMA Reset.
WDT	1	Watchdog Timer Reset.
GPIO0	2	GPIO0 Reset. Setting this bit to 1 resets GPIO0 pins to their default states.
GPIO1	3	GPIO1 Reset. Setting this bit to 1 resets GPIO1 pins to their default states.
RFU	4	Reserved
TIMER0	5	Timer0 Reset. Setting this bit to 1 resets Timer 0 blocks.
TIMER1	6	Timer1 Reset. Setting this bit to 1 resets Timer 1 blocks.
TIMER2	7	Timer2 Reset. Setting this bit to 1 resets Timer 2 blocks.
TIMER3	8	Timer3 Reset. Setting this bit to 1 resets Timer 3 blocks.
RFU	10:9	Reserved
UART0	11	UART0 Reset. Setting this bit to 1 resets all UART 0 blocks.
RFU	12	Reserved
SPI0	13	SPI0 Reset. Setting this bit to 1 resets all SPI 0 blocks.
SPI1	14	SPI1 Reset. Setting this bit to 1 resets all SPI 1 blocks.
RFU	15	Reserved
I2C0	16	I2C0 Reset.
RFU	17	Reserved
CRYPTO	18	Cryptographic Reset. Setting this bit to 1 resets the AES, SHA, and DES blocks.
RFU	28:19	Reserved
SRST	29	Soft Reset. Setting this bit to 1 resets everything except the CPU and the watchdog timer.
PRST	30	Peripheral Reset. Setting this bit to 1 resets all peripherals. The CPU core, watchdog timer, and all GPIO pins are unaffected by this reset.
SYSTEM	31	System Reset. Setting this bit to 1 resets the CPU core and all peripherals, including the watchdog timer.

4.4.3 Clock Control Register (CLKCN, Offset 0x0008)

Table 4-6. CLKCN (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[1:0]		LIRC8K_RDY	HIRC8M_RDY	RFU	HIRC_RDY	RFU[1:0]	
Reset	11		*	*	0	*	00	
Access	R		R	R	R	R	R	
Position	23	22	21	20	19	18	17	16
Field	RFU[1:0]		HIRC8M_VS	HIRC8M_EN	RFU	HIRC_EN	RFU[1:0]	
Reset	00		0	1	0	1	00	
Access	R		RW	RW	R	RW	R	
Position	15	14	13	12	11	10	9	8
Field	CCD	RFU	CKRDY	RFU	CLKSEL[2:0]			PSC2
Reset	0	0	0	0	000			0
Access	R	R	R	R	RW			RW
Position	7	6	5	4	3	2	1	0
Field	PSC1	PSC0	RFU[5:0]					
Reset	0	0	00 1000					
Access	RW	RW	R					

Name	Bits	Description	Settings
RFU	5:0	Reserved	N/A
PSC	8:6	Prescaler Select. This 3-bit field sets the system operating frequency by controlling the prescaler that divides the system clock. System clock is set by CLKSEL	000: /1 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128
CLKSEL	11:9	Clock Source Select. This 3-bit field selects the source for the system clock.	000: 120MHz 001: Reserved 010: Reserved 011: Nano-ring 100: Reserved 101: 8MHz Clock Others: Reserved. Crypto oscillator is used.
RFU	12	Reserved.	N/A
CKRDY	13	Clock Ready. This read only bit reflects whether the currently selected system clock source is running.	0: Switchover from the previously active clock source to the new clock source (as selected by CLKSEL) has not yet occurred. 1: System clock running from CLKSEL clock source.
RFU	14	Reserved	N/A
CCD	15	Cryptographic Clock Divider	0: The cryptographic accelerator clock is running in non-divided mode. 1: The cryptographic accelerator clock is running in divided mode.
RFU	16:17	Reserved.	N/A
HIRC_EN	18	120MHz High frequency internal reference clock is enabled.	0: Disabled 1: Enabled
RFU	19	Reserved	N/A

Name	Bits	Description	Settings
HIRC8M_EN	20	8MHz High Frequency Internal Reference Clock Enable	0: Disabled 1: Enabled
HIRC8M_VS	21	8MHz High Frequency Internal Reference Clock Voltage Select. This register bit is used to select the power supply to the HIRC8M. Note: Upon entering DEESLEEP mode, the oscillator supply automatically switches to the dedicated 1V regulated supply.	0: V _{CORE} supply. 1: Dedicated 1V regulated supply.
RFU	25:22	Reserved	N/A
HIRC_RDY	26	120MHz HIRC Ready	0: The HIRC is not ready. 1: The HIRC is ready.
RFU	27	Reserved	N/A
HIRC8M_RDY	28	8MHz HIRC Ready	0: The 8MHz HIRC is not ready. 1: The 8MHz HIRC is ready.
LIRC8K_RDY	29	8kHz Low Frequency Internal Reference Clock Ready	0: The 8kHz LIRC is not ready. 1: The 8kHz LIRC is ready.
RFU	31:30	Reserved.	N/A

4.4.4 Power Management Register (PM, Offset 0x000C)

Table 4-7. PM (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[5:0]						HIRC8MPD	RFU
Reset	0000 0						1	1
Access	R						RW	R
Position	15	14	13	12	11	10	9	8
Field	HIRCPD	RFU	RFU	RFU	RFU[3:0]			
Reset	1	1	1	1	0000			
Access	RW	R	R	R	R			
Position	7	6	5	4	3	2	1	0
Field	RFU	RFU	RFU	GPIOWKEN	RFU	MODE2	MODE1	MODE0
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	RW	R	RW	RW	RW

Name	Bits	Description	Settings
MODE	2:0	Operating Mode. This three-bit field selects the current operating mode for the device. Note that code execution only occurs during ACTIVE mode.	000: ACTIVE xx1: SLEEP x1x: DEEPSLEEP 1xx: BACKUP x11: SHUTDOWN
RFU	3	Reserved.	N/A
GPIOWKEN	4	GPIO Wake Up Enable. This bit enables all GPIO pins as potential wakeup sources. Any GPIO configured for wakeup can cause an exit from SLEEP, DEEPSLEEP, or BACKUP modes when this bit is set.	0: Wakeup from GPIO pin activity is disabled. 1: Wakeup is enabled. Appropriate signals on GPIO pins configured to act as wakeup sources cause the device to wake up from SLEEP, DEEPSLEEP, BACKGROUND or BACKUP modes.
RFU	14:5	Reserved.	N/A
HIRCPD	15	HIRC Power Down. This bit selects HIRC power state in DEEPSLEEP mode. This bit must be set to power down HIRC in DEEPSLEEP mode.	0: Active 1: Powered Down in DEEPSLEEP
RFU	16	Reserved	N/A
HIRC8MPD	17	8MHz HIRC Power Down. This bit selects 8MHz HIRC power state in DEEPSLEEP mode. This bit must be set to power down HIRC8M in DEEPSLEEP mode.	0: Active 1: Powered Down in DEEPSLEEP
RFU	31:18	Reserved.	N/A

4.4.5 Peripheral Clock Divider Register (PCKDIV, Offset 0x0018)

Table 4-8. PCKDIV (Offset 0x0018)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	AONCD		RFU[5:0]					
Reset	00		00 0000					
Access	RW		R					
Position	7	6	5	4	3	2	1	0
Field	RFU[4:0]				PCFWEN	PCF		
Reset	0000 0				0	001		
Access	R				RW	RW		

Name	Bits	Description	Settings
PCF	2:0	These bits determine the clock frequency for the legacy UART peripherals. These peripherals have an adaptive clock generator that dynamically adjusts the peripheral frequency based on the main system bus frequency. These bits can be written when PCFWEN is set.	2: HCLK=120MHz 3: HCLK=60MHz 4: HCLK=30MHz 5: HCLK=15MHz 6: HCLK=7.5MHz 7: HCLK=3.75MHz Others: Invalid
PCFWEN	3	PCF Write Enable. This bit allows the PCF register bits to be updated by the software.	0: Writes to PCF are blocked 1: Writes to PCF are allowed
RFU	13:4	Reserved	
AONCD	15:14	Always-ON (AON) domain Clock Divider. These bits define the AON domain clock divider.	0: HCLK divide by 4 1: HCLK divide by 8 2: HCLK divide by 16 3: HCLK divide by 32
RFU	31:16	Reserved.	N/A

4.4.6 Peripheral Clock Disable Register (PERCKCN, Offset 0x0024)

Table 4-9. PERCKCN (Offset 0x0024)

Position	31	30	29	28	27	26	25	24
Field	RFU							
Reset	1111 1111							
Access	R							

Position	23	22	21	20	19	18	17	16
Field	RFU					T3D	T2D	T1D
Reset	1111 1					1	1	1
Access	R					RW	RW	R

Position	15	14	13	12	11	10	9	8
Field	T0D	CRYPTOD	I2C0D	RFU			UART0D	RFU
Reset	1	1	1	111			1	1
Access	RW	RW	RW	R			RW	R

Position	7	6	5	4	3	2	1	0
Field	SPI1D	SPI0D	DMAD	RFU			GPIO1D	GPIO0D
Reset	1	1	1	111			1	1
Access	RW	RW	RW	R			RW	RW

Clearing a bit to 0 in this register enables the clock for the associated peripheral(s). If the bit is set to 1 (which is the default state), the peripheral clock is disabled.

Name	Bits	Description
GPIO0D	0	GPIO0 Disable
GPIO1D	1	GPIO1 Disable.
RFU	4:2	Reserved for future use.
DMAD	5	DMA Disable
SPI0D	6	SPI 0 Disable.
SPI1D	7	SPI 1 Disable.
RFU	8	Reserved for future use
UART0D	9	UART 0 Disable.
Reserved	12:10	Reserved for future use
I2C0D	13	I2C 0 Disable
CRYPTOD	14	Crypto Disable.
T0D	15	Timer 0 Disable.
T1D	16	Timer 1 Disable.
T2D	17	Timer 2 Disable.
T3D	18	Timer 3 Disable.
RFU	31:19	Reserved for future use

4.4.7 Memory Clock Control Register (MEMCKCN, Offset 0x0028)

Table 4-10. MEMCKCN (Offset 0x0028)

Position	31	30	29	28	27	26	25	24
Field	RFU[1:0]		ROMLS	RFU				ICACHELS
Reset	00		0	0 000				0
Access	R		RW	R				RW
Position	23	22	21	20	19	18	17	16
Field	RFU[2:0]			SYSRAM4LS	SYSRAM3LS	SYSRAM2LS	SYSRAM1LS	SYSRAM0LS
Reset	000			0	0	0	0	0
Access	R			RW	RW	RW	RW	RW
Position	15	14	13	12	11	10	9	8
Field	RFU							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[4:0]					FWS[2:0]		
Reset	0000 0					010		
Access	R					RW		

Name	Bits	Description	Settings
FWS	2:0	Flash Wait State. These bits define the number of wait-state cycles per Flash data read access. If the system clock is on HIRC clock, the minimum wait state is 2. Otherwise, the minimum wait state is 1 for other clock sources.	0: Invalid 1-7: number of wait states per Flash data read access.
RFU	15:3	Reserved	N/A
SYSRAM0LS	16	System RAM 0 Light Sleep mode.	0: RAM is active. 1: RAM is in Light Sleep mode.
SYSRAM1LS	17	System RAM 1 Light Sleep mode.	0: RAM is active. 1: RAM is in Light Sleep mode.
SYSRAM2LS	18	System RAM 2 Light Sleep mode.	0: RAM is active. 1: RAM is in Light Sleep mode.
SYSRAM3LS	19	System RAM 3 Light Sleep mode.	0: RAM is active. 1: RAM is in Light Sleep mode.
SYSRAM4LS	20	System RAM 4 Light Sleep mode.	0: RAM is active. 1: RAM is in Light Sleep mode.
RFU	23:21	Reserved.	N/A
ICACHELS	24	I-Cache RAM Light Sleep Mode	0: I-Cache RAM is active. 1: I-Cache RAM is in Light Sleep mode.
RFU	28:25	Reserved	N/A
ROMLS	29	ROM Light Sleep Mode	0: ROM is active 1: ROM is in Light Sleep Mode
RFU	31:30	Reserved.	N/A

4.4.8 Memory Zeroize Control Register (MEMZCN, Offset 0x002C)

Table 4-11. MEMZCN (Offset 0x002C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[6:0]							ICACHEZ
Reset	0000 000							0
Access	R							RW
Position	7	6	5	4	3	2	1	0
Field	RFU			SRAM4Z	SRAM3Z	SRAM2Z	SRAM1Z	SRAM0Z
Reset	000			0	0	0	0	0
Access	R			RW	RW	RW	RW	RW

Setting any bit to 1 initiates zeroize operation for the associated memory. This bit is automatically cleared by the hardware on zeroization completion.

Name	Bits	Description
SRAM0Z	0	System RAM block 0 Zeroization
SRAM1Z	1	System RAM block 1 Zeroization
SRAM2Z	2	System RAM block 2 Zeroization
SRAM3Z	3	System RAM block 3 Zeroization
SRAM4Z	4	System RAM block 4 Zeroization
RFU	7:5	Reserved
ICACHEZ	8	Instruction Cache Data and Tag RAM zeroization
RFU	31:9	Reserved

4.4.9 System Status Register (SYSST, Offset 0x0040)

Table 4-12. SYSST (Offset 0x0040)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							ICELOCK
Reset	0000 000							0
Access	R							R

Name	Bits	Description	Settings
ICELOCK	0	Arm ICE Lock Status	0: ICE is unlocked 1: ICE is locked
RFU	31:1	Reserved	N/A

4.4.10 Reset 1 Register (RSTR1, Offset 0x0044)

Table 4-13. RSTR1 (Offset 0x0044)

Position	31	30	29	28	27	26	25	24
Field	RFU[2:0]			SFES	RFU[3:0]			
Reset	000			0	0000			
Access	R			RS	R			
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[6:0]							WDT1
Reset	0000 000							0
Access	R							RS
Position	7	6	5	4	3	2	1	0
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							

Setting a bit to 1 in this register resets the corresponding peripheral or set of peripherals. Once the reset has completed, the bit is automatically cleared to 0 by the hardware. The software writes of 0 to bits in this register have no effect.

Name	Bits	Description
RFU	7:0	Reserved for future use
WDT1	8	Watchdog Timer 1 Reset
RFU	27:9	Reserved for future use
SFES	28	Serial Flash Emulation Slave Reset
RFU	31:29	Reserved for future use.

4.4.11 Peripheral Clock Disable Register 1 (PERCKCN1, Offset 0x0048)

Table 4-14. PERCKCN1 (Offset 0x0048)

Position	31	30	29	28	27	26	25	24
Field	RFU	SFES	RFU[1:0]		WDT1	WDT0	RFU[1:0]	
Reset	1	1	11		11	1	11	
Access	R	RW	R		RW	RW	R	

Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	1111 1111							
Access	R							

Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				RFU	RFU[2:0]		
Reset	1111				0	111		
Access	R				R	R		

Position	7	6	5	4	3	2	1	0
Field	RFU[4:0]					TRNGD	RFU	
Reset	1111 1					1	11	
Access	R					RW	R	

Name	Bits	Description	Settings
RFU	1:0	Reserved	N/A
TRNGD	2	TRNG Clock Disable	0: Enabled 1: Disabled
RFU	25:3	Reserved	N/A
WDT0	26	Watchdog Timer 0 Disable	0: Enabled 1: Disabled
WDT1	27	Watchdog Timer 1 Disable	0: Enabled 1: Disabled
RFU	29:28	Reserved	N/A
SFES	30	Serial Flash Emulation Slave Disable	0: Enabled 1: Disabled
RFU	31	Reserved.	N/A

4.4.12 Event Enable Register (EVTEN, Offset 0x004C)

Table 4-15. EVTEN (Offset 0x004C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[4:0]					TXEVENT	RFU	DMAEVENT
Reset	0000 0					0	0	0
Access	R					RW	R	RW

Name	Bits	Description	Settings
DMAEVENT	0	DMA CTZ Event Wake-Up Enable. When set, when a DMA block transfer is completed and the DMA counter DMA _n _CNT.cnt = 0, a CTZ DMA event occurs which generates an RXEV to wake-up the CPU from a WFE instruction.	0: Disabled 1: Enabled
RFU	1	Reserved	
TXEVENT	2	Transmit Event (TXEV) Pin Enable When set, an SEV (Send Event) instruction outputs a single-cycle pulse on output pin TXEV. TXEV is Alternate Function 1 (AF1) on pin GPIO0.25 on this device. For proper operation, set txevent = 1 only when this pin is configured for AF1.	0: Disabled 1: Enabled
RFU	31:3	Reserved	N/A

4.4.13 Revision Register (REVISION, Offset 0x0050)

Table 4-16. REVISION (Offset 0x0050)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	REVISION[15:8]							
Reset	Current Chip Revision							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	REVISION[7:0]							
Reset	Current Chip Revision							
Access	R							

Name	Bits	Description	Settings
REVISION	15:0	Chip Revision	A1 A2 ... B1 B2 ... etc.
RFU	31:16	Reserved	N/A

4.4.14 System Status Interrupt-Enabled Register (SYSSIE, Offset 0x0054)

Table 4-17. SYSSIE (Offset 0x0054)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							ICEULIEK
Reset	0000 000							0
Access	R							RW

Name	Bits	Description	Settings
ICEULIE	0	Arm ICE Unlock Interrupt Enable	0: Disabled 1: Enabled
RFU	31:1	Reserved	N/A

4.4.15 ECC Error Register (ECCERR, Offset 0x0064)

Table 4-18. ECCERR (Offset 0x0064)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			ECCSYSRAM4 ERR	ECCSYSRAM3 ERR	ECCSYSRAM2 ERR	ECCSYSRAM1 ERR	ECCSYSRAM0 ERR
Reset	000			0	0	0	0	0
Access	R			W1C,R	W1C,R	W1C,R	W1C,R	W1C,R

Name	Bits	Description	Settings
ECCSYSRAM0 ERR	0	ECC System RAM0 Error Flag. Write 1 to clear.	
ECCSYSRAM1 ERR	1	ECC System RAM1 Error Flag. Write 1 to clear.	
ECCSYSRAM2 ERR	2	ECC System RAM2 Error Flag. Write 1 to clear.	
ECCSYSRAM3 ERR	3	ECC System RAM3 Error Flag. Write 1 to clear.	
ECCSYSRAM4 ERR	4	ECC System RAM4 Error Flag. Write 1 to clear.	
RFU	5:31	Reserved	N/A

4.4.16 ECC Not Double Error Detect Register (ECCNDED, Offset 0x0068)

Table 4-19. ECCNDED (Offset 0x0068)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[3:0]			ECCSYSRAM4 NDED	ECCSYSRAM3 NDED	ECCSYSRAM2 NDED	ECCSYSRAM1 NDED	ECCSYSRAM0 NDED
Reset	000			0	0	0	0	0
Access	R			W1C,R	W1C,R	W1C,R	W1C,R	W1C,R

Name	Bits	Description	Settings
ECCSYSRAM0NDED	0	ECC System RAM0 Not Double Error Detect. Write 1 to clear.	
ECCSYSRAM1NDED	1	ECC System RAM1 Not Double Error Detect. Write 1 to clear.	
ECCSYSRAM2NDED	2	ECC System RAM2 Not Double Error Detect. Write 1 to clear.	
ECCSYSRAM3NDED	3	ECC System RAM3 Not Double Error Detect. Write 1 to clear.	
ECCSYSRAM4NDED	4	ECC System RAM4 Not Double Error Detect. Write 1 to clear.	
RFU	31:5	Reserved.	N/A

4.4.17 ECC Interrupt Enable Register (ECC_INT_EN, Offset 0x006C)

Table 4-20. ECC_INT_EN (Offset 0x006C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			ECCSYSRAM4 INTEN	ECCSYSRAM3 INTEN	ECCSYSRAM2 INTEN	ECCSYSRAM1 INTEN	ECCSYSRAM0 INTEN
Reset	000			0	0	0	0	0
Access	R			RW	RW	RW	RW	RW

Name	Bits	Description	Settings
ECCSYSRAM0INTEN	0	System RAM0 ECC Error Interrupt Enable	0: Disable 1: Enable
ECCSYSRAM1INTEN	1	System RAM1 ECC Error Interrupt Enable	0: Disable 1: Enable
ECCSYSRAM2INTEN	2	System RAM2 ECC Error Interrupt Enable	0: Disable 1: Enable
ECCSYSRAM3INTEN	3	System RAM3 ECC Error Interrupt Enable	0: Disable 1: Enable
ECCSYSRAM4INTEN	4	System RAM4 ECC Error Interrupt Enable	0: Disable 1: Enable
RFU	5:31	Reserved	N/A

4.4.18 ECC Enable Register (ECCEN, Offset 0x0000)

Table 4-21. ECCEN (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			ECCSYSRAM4 EN	ECCSYSRAM3 EN	ECCSYSRAM2 EN	ECCSYSRAM1 EN	ECCSYSRAM0 EN
Reset	000			0	0	0	0	0
Access	R			RW	RW	RW	RW	RW

Name	Bits	Description	Settings
ECCSYSRAM0EN	0	ECC System RAM0 Enable	0: Disable 1: Enable
ECCSYSRAM1EN	1	ECC System RAM1 Enable	0: Disable 1: Enable
ECCSYSRAM2EN	2	ECC System RAM2 Enable	0: Disable 1: Enable
ECCSYSRAM3EN	3	ECC System RAM3 Enable	0: Disable 1: Enable
ECCSYSRAM4EN	4	ECC System RAM4 Enable	0: Disable 1: Enable
RFU	5:31	Reserved	N/A

4.5 Low-Power Management Register

This set of registers control the Low Power Management of the MAX32520. Address assignments for the Security Monitor registers are outlined in Table 4-22.

Table 4-22. Low -Power Management Addresses (Base ADDR = 0x4000_6800)

Offset	Access	Register	Description
0x0000	RW	LPNC	Low Power Control Register
0x0004	RW	LPWKST0	Low Power I/O Wakeup Status Register for GPIO0
0x0008	RW	LPWKEN0	Low Power I/O Wakeup Enable Register for GPIO0
0x000C	RW	LPWKST1	Low Power I/O Wakeup Status Register for GPIO1
0x0010	R	LPWKEN1	Low Power I/O Wakeup Enable Register for GPIO1
0x0014 - 0x0028	R	RFU	Reserved for future use
0x0030	RW	LPPWST	Low Power Peripheral Wakeup Status Register
0x0034 - 0x003C	RW	RFU	Reserved for future use
0x0040	RW	LPMEMSD	Low Power Memory Shutdown Control
0x0044	R	RFU	Reserved for future use
0x0048	RW	BUGPR0	Backup Mode General Purpose Register 0
0x004C	RW	BUGPR1	Backup Mode General Purpose Register 1

4.5.1 Low-Power Control Register (LPCN, Offset 0x0000)

Table 4-23. LPCN (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU	VDDAMD	RFU	VDDCMD	RFU[3:0]			LDOLVE
Reset	0	0	0	0	000			0
Access	R	RW	R	RW	R			RW
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[4:0]					RAMRET2	RAMRET1	RAMRET0
Reset	0000 0					0	0	0
Access	R					RW	RW	RW

Name	Bits	Description	Settings
RAMRET[2:0]	2:0	System RAM retention in BACKUP mode.	000: Disable RAM retention xx1: Enable System RAM 0 retention x1x: Enable System RAM 1 retention 1xx: Enable System RAM 4 retention
RFU	15:3	Reserved. Must be written with zeros for future compatibility.	N/A
LDOLVE	16	LDO Low Voltage Enable. Setting this bit enables the LDO low voltage mode.	0: Enabled 1: Disabled (default)
RFU	19:17	Reserved. Must be written with zeros for future compatibility.	N/A
VDDCMD	20	V _{DDC} (V _{CORE}) Monitor Disable. This bit controls the power monitor on the V _{CORE} supply in all operating modes.	0: Enabled if Bandgap is ON (default) 1: Disabled
RFU	21	Reserved.	N/A
VDDAMD	22	V _{DDA} Monitor Disable. This bit controls the power monitor on the analog supply in all operating modes.	0: Enabled if Bandgap is ON (default) 1: Disabled
RFU	31:23	Reserved.	N/A

4.5.2 Low-Power I/O Wakeup Status Register 0 (LPWKST0, Offset 0x0004)

Table 4-24. LPWKST0 (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	WAKEST[15:8]							
Reset	0000 0000							
Access	RW Write 1 to Clear							
Position	7	6	5	4	3	2	1	0
Field	WAKEST[7:0]							
Reset	0000 0000							
Access	RW Write 1 to Clear							

Name	Bits	Description
WAKEST	15:0	Wakeup IRQ flags (write ones to clear). One or more of these bits are set when the corresponding dedicated GPIO0 pin(s) transition(s) from low to high or high to low. If GPIO wakeup source is selected, using PM.GPIOWKEN register, and the corresponding bit is also selected in the LPWKEN0 register, an interrupt is generated to wake up the CPU from a low-power mode.
Reserved	31:16	Reserved.

4.5.3 Low-Power I/O Wakeup Enable Register 0 (LPWKEN0, Offset 0x0008)

Table 4-25. LPWKEN0 (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	WAKEEN[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	WAKEEN[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description
WAKEEN	15:0	Enable wakeup. These bits allow wakeup from the corresponding GPIO0 pin(s) on transition(s) from low to high or high to low when PM.GPIOWKEN is set. The wakeup status is indicated in the LPWKST0 register.
Reserved	31:16	Reserved.

4.5.4 Low-Power I/O Wakeup Status Register 1 (LPWKST1, Offset 0x000C)

Table 4-26. LPWKST1 (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[4:0]					WAKEST[10:8]		
Reset	0000 0					000		
Access	R					RW Write 1 to Clear		
Position	7	6	5	4	3	2	1	0
Field	WAKEST[7:0]							
Reset	0000 0000							
Access	RW Write 1 to Clear							

Name	Bits	Description
WAKEST	10:0	Wakeup IRQ flags (write ones to clear). One or more of these bits are set when the corresponding dedicated GPIO1 pin(s) transition(s) from low to high or high to low. If GPIO wakeup source is selected, using PM.GPIOWKEN register, and the corresponding bit is also selected in the LPWKEN1 register, an interrupt is generated to wake up the CPU from a low-power mode.
Reserved	31:11	Reserved.

4.5.5 Low-Power I/O Wakeup Enable Register 1 (LPWKEN1, Offset 0x0010)

Table 4-27. LPWKEN1 (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[4:0]					WAKEEN[15:8]		
Reset						000		
Access						RW		
Position	7	6	5	4	3	2	1	0
Field	WAKEEN[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description
WAKEEN	10:0	Enable wakeup. These bits allow wakeup from the corresponding GPIO1 pin(s) on transition(s) from low to high or high to low when PM.GPIOWKEN is set. The wakeup status is indicated in the LPWKST1 register.
Reserved	31:11	Reserved.

4.5.6 Low-Power Peripheral Wakeup Status Register (LPPWST, Offset 0x0030)

Table 4-28. LPPWST (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[4:0]					DRSWKST	RSTWKST	BBMOSEDT
Reset	0000 0					0	0	0
Access	R					RW1C	RW1C	RW1C
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description
RFU	15:0	Reserved
BBMODEST	16	Wakeup Flag from Backup mode status (write one to clear)
RSTWKST	17	Reset Detect Wakeup Flag (write one to clear). This bit is set when the external reset causes wakeup condition
DRSWKST	18	DRS Detect Wakeup Flag (write one to clear).
RFU	31:17	Reserved

4.5.7 Low-Power Memory Shutdown Register (LPMEMSD, Offset 0x0040)

Table 4-29. LPMEMSD (Offset 0x0040)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[2:0]			ROMSD	RFU[3:0]			
Reset	000			0	0000			
Access	R			RW	R			
Position	7	6	5	4	3	2	1	0
Field	ICACHESD	RFU[1:0]		SYSRAM4SD	SYSRAM3SD	SYSRAM2SD	SYSRAM1SD	SYSRAM0SD
Reset		00		0	0	0	0	0
Access		R		RW	RW	RW	RW	RW

Name	Bits	Description	Settings
SYSRAM0SD	0	System RAM block 0 Shut Down.	0: Normal operating mode 1: Shutdown mode
SYSRAM1SD	1	System RAM block 1 Shut Down.	0: Normal operating mode 1: Shutdown mode
SYSRAM2SD	2	System RAM block 2 Shut Down.	0: Normal operating mode 1: Shutdown mode
SYSRAM3SD	3	System RAM block 3 Shut Down.	0: Normal operating mode 1: Shutdown mode
SYSRAM4SD	4	System RAM block 4 Shut Down.	0: Normal operating mode 1: Shutdown mode
RFU	6:5	Reserved.	N/A
ICACHESD	7	Instruction Cache RAM Shut Down.	0: Normal operating mode 1: Shutdown mode
RFU	11:8	Reserved	N/A
ROMSD	12	ROM Shut Down.	0: Normal operating mode 1: Shutdown mode
RFU	31:13	Reserved	N/A

4.5.8 Backup Mode General Register 0 (BUGPR0, Offset 0x0048)

Table 4-30. BUGPR0 (Offset 0x0048)

Position	31	30	29	28	27	26	25	24
Field	BUGPR0[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	BUGPR0[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	BUGPR0[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	BUGPR0[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
BUGPR0	31:0	Backup general purpose register 0. This register is used to store the return address when exiting from the Backup mode.	

4.5.9 Backup Mode General Register 1 (BUGPR1, Offset 0x004C)

Table 4-31. BUGPR1 (Offset 0x004C)

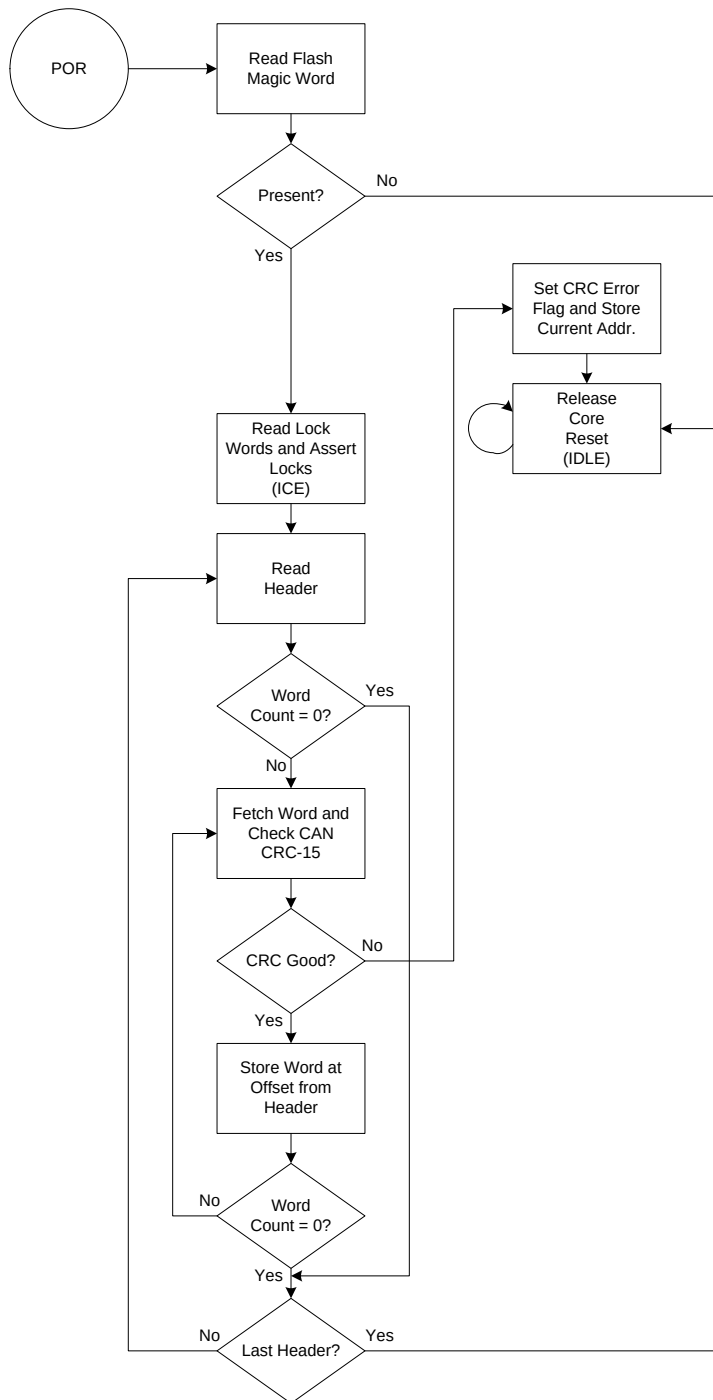
Position	31	30	29	28	27	26	25	24
Field	BUGPR1[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	BUGPR1[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	BUGPR1[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	BUGPR1[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
BUGPR1	31:0	Backup general purpose register 1. This register is used to store the return address when exiting from the Backup Mode.	

5.0 System Initialization

Following initial power-up, the device performs certain actions to initialize its starting state before it begins to execute application code. These actions include retrieving information from the nonvolatile memory to configure the operation of the device. This non-volatile data (NVD) includes analog trim information, configuration, and locking bits, and may also include customer- and application-specific information such as serial numbers and cryptographic keys. The system initialization startup flow is shown in Figure 5-1.

Figure 5-1. System Initialization Block Startup Flow



5.1 System Initialization Block (SIB)

The System Initialization Block (or SIB) is a functional block on the device responsible for performing initialization functions following initial power-up. The SIB is activated automatically following a power-on reset, and it holds the main CPU in a reset state while it performs its initialization functions.

While the CPU is held in a reset state, the SIB takes control of the AHB bus and reads trim and configuration data from the on-chip one-time-programmable (OTP) memory. This memory is loaded with appropriate values during the manufacturing and test cycle. The SIB reads values from the OTP block and loads trim registers and other configuration registers as appropriate. Once the initialization process is complete, the SIB disengages from the AHB bus and releases the CPU core from reset, allowing it to begin code execution.

5.2 System Initialization Registers (SIR)

In general, the SIB block performs initialization of the device automatically, and the user need not be concerned with the exact details of which values are in the OTP memory block and how they are interpreted. The only potential issue comes if the OTP values have not been loaded, or if the OTP values are misread or somehow corrupted. In this case, certain registers in the SIR register area may be checked by application code to verify the successful completion of the system initialization process.

Address assignments for the System Initialization registers are outlined in Table 5-1.

Table 5-1. System Initialization Register Addresses (Base ADDR = 0x4000_0400)

Offset	Access	Register	Description
0x0000	R	SISTAT	System Initialization Status Register
0x0004	R	ERRADDR	System Initialization Error Address Register
0x0100	R	FNSTAT	Function Status Register
0x0104	R	SFNSTAT	Security Function Status Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on reset values and various access modes.

5.2.1 System Initialization Status Register (SISTAT, Offset 0x0000)

Table 5-2. SISTAT (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[5:0]						CRCERR	MAGIC
Reset	0000 00						0	0
Access	R						R	R

Name	Bits	Description	Settings
MAGIC	0	Read-only status bit set by the system initialization block following power-up.	0: Magic word was not set (OTP has not been initialized properly). 1: Magic word was set – OTP contains valid settings.
CRCERR	1	Read-only status bit set by the system initialization block following power-up.	0: No CRC errors occurred during the read of the OTP memory block 1: A CRC error occurred while reading the OTP. The address of the failure location in the OTP memory is stored in the ERRADDR register.
RFU	31:2	Reserved	N/A

5.2.2 System Initialization Error Address Register (ERRADDR, Offset 0x0004)

Table 5-3. ERRADDR (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	ERRADDR[31:24]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	ERRADDR[23:16]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	ERRADDR[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	ERRADDR[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
ERRADDR	31:0	Read-only field set by the SIB block if a CRC error occurs during the read of the OTP memory.	Contains the failing address in OTP memory (when CRCERR equals 1).

5.2.3 Function Status Register (FNSTAT, Offset 0x0100)

Table 5-4. FNSTAT (Offset 0x0100)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[1:0]		RFU	SFES	NMI	RFU[2:0]		
Reset	00		0	*	*	000		
Access	R		R	R	R	R		
Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							FPU
Reset	0000 000							*
Access	R							R

Name	Bits	Description	Settings
FPU	0	Floating point function	0: Not Available 1: Available
RFU	10:1	Reserved	N/A
NMI	11	NMI interrupt is enabled	0: No 1: Yes
SFES	12	SFES Slave function	0: Not Available 1: Available
RFU	13	Reserved	N/A
RFU	31:14	Reserved	N/A

5.2.4 Security Function Status Register (SFNSTAT, Offset 0x0104)

Table 5-5. SFNSTAT (Offset 0x0104)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			SHA	AES	TRNG	RFU[1:0]	
Reset	00x			*	*	*	00	
Access	R			R	R	R	R	

Name	Bits	Description	Settings
RFU	1:0	Reserved	N/A
TRNG	2	TRNG function	0: Not Available 1: Available
AES	3	AES function	0: Not Available 1: Available
SHA	4	SHA function	0: Not Available 1: Available
RFU	31:5	Reserved	N/A

6.0 Interrupts

The following table shows the Interrupt Vector for each peripheral.

Table 6-1. Interrupt Vectors

Interrupt Number	Offset	Interrupt Source	Interrupt Number	Offset	Interrupt Source
16	0040	Power Fail Interrupt	41	00A4	GPIO 1
17	0044	Watchdog Timer	42	00A8	Reserved
18	0048	Reserved	43	00AC	Crypto Engine
19	004C	Reserved	44	00B0	DMA 0
20	0050	TRNG	45	00B4	DMA 1
21	0054	Timer 0	46	00B8	DMA 2
22	0058	Timer 1	47	00BC	DMA 3
23	005C	Timer 2	48-69	00C0-0114	Reserved
24	0060	Timer 3	70	0118	GPIO/Peripheral wake up
25	0064	Reserved	71-72	011C-0120	Reserved
26	0068	Reserved	73	0124	Watchdog timer 1
27	006C	Reserved	74-97	0128-0184	Reserved
28	0070	Reserved	98	0188	Error Correction
29	0074	I2C 0	99-100	018C-0190	Reserved
30	0078	UART 0	101	0194	SCA Crypto Accelerator
31	007C	Reserved	102-110	0198-01B8	Reserved
32	0080	SPI 0	111	01BC	Reserved
33	0084	SPI 1	112	01C0	Serial Flash Emulation
34	0088	Reserved			
35	008C	Reserved			
36	0090	Reserved			
37	0094	Reserved			
38	0098	Reserved			
39	009C	Flash Controller			
40	00A0	GPIO 0			

Note: The interrupt vector table should be 256-byte (64 word) aligned. Otherwise, an incorrect handler may be vectored to upon an interrupt. Refer to the Arm Cortex-M4F documentation for further information on interrupt vector table alignment. For MAX32520, there are 39 interrupts and 16 Cortex-M4F exceptions. The total table size is 56 and should only need 64-word alignment.

6.1 Nested Vectored Interrupt Controller

Refer to Arm Cortex-M4F Technical Reference Manual for Nested Vectored Interrupt Controller available at www.arm.com

7.0 Instruction Cache Controller

Please refer to the Instruction Cache Controller register section.

7.1 Instruction Cache Controller Registers

Address assignments for the Instruction Cache Controller Registers are outlined in Table 7-1.

Table 7-1. Instruction Cache Controller Register Address (Base ADDR = 0x4002_A000)

Offset	Access	Register	Description
0x0000	R	CACHE_ID	Cache ID Register
0x0004	RW	MEMCFG	Memory Configuration Register
0x0008 ... 0x00FC	R	RFU	
0x0100	RW	CACHE_CTRL	Cache Control and Status Register
0x0104 ... 0x06FC	R	RFU	
0x0700	R	INVALIDATE	Invalidate All Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on reset values and various access modes.

7.1.1 Cache ID Register (CACHE_ID, Offset 0x0000)

Table 7-2. CACHE_ID (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	CCHID[5:0]						PARTNUM[3:2]	
Reset	00 0000						00	
Access	R						R	
Position	7	6	5	4	3	2	1	0
Field	PARTNUM[1:0]		RELNUM[5:0]					
Reset	00		00 0000					
Access	R		R					

Name	Bits	Description	Settings
RELNUM	5:0	Release Number. Identifies the RTL release version.	
PARTNUM	9:6	Part Number. This field reflects the value of C_ID_PART_NUMBER configuration parameter.	
CCHID	15:10	Cache ID. This field reflects the value of the C_ID_CACHEID configuration parameter.	
RFU	31:16	Reserved	N/A

7.1.2 Memory Configuration Register (MEMCFG, Offset 0x0004)

Table 7-3. MEMCFG (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	MEMSZ[15:8]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	MEMSZ[7:0]							
Reset	0000 1000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	CCHSZ[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	CCHSZ[7:0]							
Reset	0000 1000							
Access	R							

Name	Bits	Description	Settings
CCHSZ	15:0	Cache Size. Indicates total size in Kbytes of cache	
MEMSZ	31:16	Main Memory Size. Indicates the total size, in units of 128 Kbytes, of code memory accessible to the cache controller	

7.1.3 Cache Control and Status Register (CACHE_CTRL, Offset 0x0100)

Table 7-4. CACHE_CTRL (Offset 0x0100)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[6:0]							CACHE_RDY
Reset	0000 000							0
Access	R							RW
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							CACHE_EN
Reset	0000 000							0
Access	R							RW

Name	Bits	Description	Settings
CACHE_EN	0	Cache Enable. Controls if the cache is bypassed or is in use. Changing the state of this bit causes the instruction cache to be flushed and its contents invalidated.	0: Cache Bypassed. Instruction data is stored in the line fill buffer but is not written to the main cache memory array. 1: Cache enabled
RFU	15:1	Reserved	N/A
CACHE_RDY	16	Cache Ready flag. Cleared by hardware at any time the cache is invalidated (including a system reset). When this bit is 0, the cache is effectively in bypass mode (instruction fetches come from main memory or from the line fill buffer). Set by hardware when the invalidate operation is complete and the cache is ready.	
RFU	31:17	Reserved	N/A

7.1.4 Invalidate All (INVALIDATE, Offset 0x0700)

Table 7-5. INVALIDATE (Offset 0x0700)

Position	31	30	29	28	27	26	25	24
Field	IA[31:24]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	IA[23:16]							
Reset	000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	IA[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	IA[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
IA	31:0	Invalidate All Cache Contents. Any time this register location is written (regardless of the data value), the cache controller immediately begins invalidating the entire contents of the cache memory. The cache is in bypass mode until the invalidate operation is complete. The system software can examine the Cache Ready bit (CACHE_CTRL.CACHE_RDY) to determine when the invalidate operation is complete. Note that it is not necessary to disable the cache controller prior to beginning this operation. Reads from this register always return 0.	

8.0 Embedded ICE and Debug

This device has a JTAG interface as well as Single Wire Interface (SWD) and an embedded In-Circuit Emulator (ICE) interface. Refer to the Arm Cortex-M4F Technical Reference Manual for Debug available at www.arm.com

9.0 General-Purpose Input/Output (GPIO) Pins

General-Purpose I/O (GPIO) pins can be individually configured to operate in a digital I/O mode or in an alternate function (AF) mode, which maps a signal associated with an enabled peripheral to that GPIO. The number of GPIO pins and the assignment of alternate functions are shown in the GPIO and Alternate Function Matrices. The GPIO supports dynamic switching between the I/O mode and its alternate function modes. Configuring a pin for an alternate function supersedes its use as a digital I/O. However, the state of the GPIO can still be read via the GPIO input register.

The electrical characteristics of a GPIO pin are identical whether the pin is configured as an I/O or alternate function, except where explicitly noted in the data sheet electrical characteristics tables.

GPIOs are logically divided into ports of 32 pins. Some devices and package variants may not implement all pins of a specific 32-bit GPIO port.

Each pin of a port has an interrupt function that can be independently enabled and configured as a level- or edge-sensitive interrupt. All GPIOs of a given port share the same interrupt vector as detailed in the section GPIO Interrupt Handling.

The features for each GPIO pin include:

- Full CMOS outputs with configurable drive strength settings.
- Input mode options:
 - High impedance
 - Weak pullup/pulldown
 - Strong pullup/pulldown
- Output data can be from the GPIO_OUT register or an enabled AF peripheral.
- Input data can be read from the GPIO_IN input register or the enabled peripheral.
- Bit set and clear registers for efficient bit-wise write access to the pins and configuration registers.
- Wake from low-power modes using edge triggered inputs.
- Selectable interrupt events:
 - Level triggered low
 - Level triggered high
 - Edge triggered rising edge
 - Edge triggered falling edge
 - Edge triggered rising or falling edge
- All GPIO pins default to input mode with weak pullup during power-on-reset events.

9.1 Usage

9.1.1 Reset State

During a power-on-reset event, each GPIO is reset to the default hi-z input mode as follows:

- The GPIO Configuration Enable bits are set to I/O mode.
- Input mode is enabled (GPIO_IN_EN[pin] = 1).
- High impedance mode enabled (GPIO_PDPU_SEL0[pin] = 0, GPIO_PDPU_SEL1[pin] = 0), pullup and pulldown disabled.
- Output mode disabled (GPIO_OUT_EN[pin] = 0).
- Interrupt disabled (GPIO_INT_EN[pin] = 0).

9.1.2 Input Mode Configuration

Perform the following steps to configure one or more pins for input mode:

1. Set the GPIO Configuration Enable bits shown in Table 9-3 according to the data sheet to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired as shown in Table 9-1.
3. Enable the input buffer connection to the GPIO pin by setting GPIO_IN_EN[pin] to 1.

4. Read the input state of the pin using the GPIO_IN[pin] field.

9.1.3 Output Mode Configuration

Perform the following steps to configure a pin for output mode:

1. Set the GPIO Configuration Enable bits shown in Table 9-3 to any one of the I/O settings.
2. Configure the electrical characteristics of the pin as desired as shown in Table 9-2.
3. Set the output high or low using the GPIO_OUT[pin] bit.
4. Enable the output buffer for the pin by setting GPIO_OUT_EN[pin] to 1.

9.1.4 Alternate Function Configuration

Most GPIOs support one or more alternate functions selected with the GPIO Configuration Enable bits shown in Table 9-3. The bits that select the AF must only be changed while the pin is in one of the I/O modes (GPIO_EN0 = 1). The specific I/O mode must match the desired AF.

1. Set the GPIO Configuration Enable bits shown in Table 9-3 to the I/O mode that corresponds with the desired new AF setting. Switching between different I/O mode settings do not affect the state or electrical characteristics of the pin.
2. Configure the electrical characteristics of the pin. See Table 9-1 if the assigned alternate function uses the pin as an input. See Table 9-2 if the assigned alternate function uses the pin as an output.
3. Set the GPIO Configuration Enable bits shown in Table 9-3 to the desired alternate function.

9.2 Configuring GPIO (External) Interrupts

Each GPIO pin supports external interrupt events when the GPIO is configured for the I/O mode and the input mode is enabled. If the GPIO is configured as a peripheral alternate function, the interrupts are peripheral-controlled.

GPIO interrupts can be individually enabled and configured as an edge or level triggered independently on a pin-by-pin basis. The edge trigger can be a rising, falling, or both transitions.

Each GPIO pin has a dedicated status bit in its corresponding GPIO_INT_STAT register. A GPIO interrupt occurs when the a status bit transitions from 0 to 1 if the corresponding bit is set in the corresponding GPIO_INT_EN register. Note that the interrupt status bit is always set when the current interrupt configuration event occurs, but an interrupt is generated only if explicitly enabled.

The following procedure details the steps for enabling the ACTIVE mode interrupt events for a GPIO pin:

1. Disable the interrupts by setting the GPIO_INT_EN[pin] field to 0. This prevents any new interrupts on the pin from triggering but not clear previously triggered (pending) interrupts. The application can disable all interrupts for a GPIO port by writing 0 to the GPIO_IN_EN register. To maintain previously enabled interrupts, read the GPIO_IN_EN register and save the state prior to setting the register to 0.
2. Clear pending interrupts by writing 1 to the GPIO_INT_CLR[pin] bit.
3. Configure the pin for the desired interrupt event
4. Set GPIO_INT_MODE[pin] to select the desired interrupt.
 - a. For level triggered interrupts, the interrupt triggers on an input high (GPIO_INT_POL[pin] = 0) or input low level.
 - b. For edge triggered interrupts, the interrupt triggers on a transition from low to high (GPIO_INT_POL[pin] = 0) or high to low (GPIO_INT_POL[pin] = 1)
5. Optionally, set GPIO_INT_DUAL_EDGE [pin] to 1 to trigger on both the rising and falling edges of the input signal.
6. Set GPIO_INT_EN[pin] to 1 to enable the interrupt for the pin.

9.2.1 GPIO Interrupt Handling

Each GPIO port is assigned its own dedicated interrupt vector as shown in Table 6-1.

To handle GPIO interrupts in your interrupt vector handler, complete the following steps:

1. Read the GPIO_INT_STAT register to determine the GPIO pin that triggered the interrupt.
2. Complete the interrupt tasks associated with the interrupt source pin (application defined).
3. Clear the interrupt flag in the GPIO_INT_STAT register by writing a 1 to the GPIO_INT_CLR bit position that triggered the interrupt. This also clears and rearms the edge detectors for edge-triggered interrupts.
4. Signal an end-of-interrupt to the interrupt controller by writing to the End-of-Interrupt register.
5. Return from the interrupt vector handler.

9.2.2 Using GPIO for Wakeup from Low-Power Modes

Low-power modes support an asynchronous wakeup from edge triggered interrupts on the GPIO ports. Level-triggered interrupts are not supported for wakeup because the system clock must be active to detect the levels. A single wakeup interrupt vector is assigned for all pins of all GPIO ports. When the GPIO wakeup event occurs, the application software must interrogate each GPIO_INT_STAT register to determine which external port pin caused the wake-up event.

To enable low power mode wakeup (SLEEP, DEEPSLEEP and BACKUP) using an external GPIO interrupt, complete the following steps:

1. Clear the pending interrupt flags by writing to GPIO_INT_CLR[pin].
2. Activate the GPIO wakeup function by writing 1 to GPIO_WAKE_EN[pin].
3. Configure the power manager to use the GPIO as a wakeup source by setting the appropriate PM.GPIOWKEN field to 1.

Table 9-1. Input Mode Configurations

Input Mode	Mode Select		Pullup/Pulldown Strength
	GPIO_PDPUS_SEL1	GPIO_PDPUS_SEL0	GPIO_PSEL
High Impedance	0	0	N/A
Weak Pullup to V _{DDIO} (1MΩ)	0	1	0
Strong Pullup to V _{DDIO} (25kΩ)	0	1	1
Weak Pulldown to V _{DDIO} (1MΩ)	1	0	0
Strong Pulldown to V _{DDIO} (25kΩ)	1	0	1
Reserved	1	1	N/A

Table 9-2. Drive Strength Configurations

Output Mode	Drive Strength	
	GPIO_DS_SEL1	GPIO_DS_SEL0
Output Drive Strength 0 (1mA Typical)	0	0
Output Drive Strength 1 (2mA Typical)	0	1
Output Drive Strength 2 (4mA Typical)	1	0
Output Drive Strength 3 (8mA Typical)	1	1

Table 9-3. GPIO Pin Configurations

Mode	Register Settings	
	GPIO_EN0	GPIO_EN1
AF1	0	0
AF2	0	1
I/O (transition to AF1)	1	0
I/O (transition to AF2)	1	1

9.3 GPIO Registers

Address assignments for GPIO registers within each Port block are outlined in Table 9-4. Reserved register bits should only be written as 0.

Table 9-4. GPIO Register Addresses (for each port)

Offset	Access	Register	Description
0x0000	RW	GPIO_EN0	GPIO Configuration Enable Bit 0 Register
0x0004	RS	GPIO_EN0_SET	GPIO Enable Bit 0 Atomic Set Register
0x0008	RS	GPIO_EN0_CLR	GPIO Enable Bit 0 Atomic Clear Register
0x000C	RW	GPIO_OUT_EN	GPIO Output Enable Register
0x0010	RS	GPIO_OUT_EN_SET	GPIO Output Enable Atomic Set Register
0x0014	RS	GPIO_OUT_EN_CLEAR	GPIO Output Enable Atomic Clear Register
0x0018	RW	GPIO_OUT	GPIO Output Register
0x001C	RS	GPIO_OUT_SET	GPIO Output Atomic Set Register
0x0020	RS	GPIO_OUT_CLR	GPIO Output Atomic Clear Register
0x0024	R	GPIO_IN	GPIO Input Register
0x0028	RW	GPIO_INT_MOD	GPIO Interrupt Mode Register
0x002C	RW	GPIO_INT_POL	GPIO Interrupt Polarity Register
0x0030	RW	GPIO_IN_EN	GPIO Input Enable Register
0x0034	RW	GPIO_INT_EN	GPIO Interrupt Enable Register
0x0038	RS	GPIO_INT_EN_SET	GPIO Interrupt Enable Atomic Set Register
0x003C	RS	GPIO_INT_EN_CLR	GPIO Interrupt Enable Atomic Clear Register
0x0040	R	GPIO_INT_STAT	GPIO Interrupt Status Register
0x0048	RS	GPIO_INT_CLR	GPIO Interrupt Status Clear
0x004C	RW	GPIO_WAKE_EN	GPIO Low Power Wake Enable Register
0x0050	RS	GPIO_WAKE_EN_SET	GPIO Low Power Wake Enable Atomic Set Register
0x0054	RS	GPIO_WAKE_EN_CLR	GPIO Low Power Wake Enable Atomic Clear Register
0x005C	RW	GPIO_INT_DUAL_EDGE	GPIO Interrupt Dual Edge Mode Register
0x0060	RW	GPIO_PDPU_SEL0	GPIO Input Mode Selection Register 0
0x0064	RW	GPIO_PDPU_SEL1	GPIO Input Mode Selection Register 1
0x0068	RW	GPIO_EN1	GPIO Configuration Enable Bit 1 Register
0x006C	RS	GPIO_EN1_SET	GPIO Enable Bit 1 Atomic Set Register
0x0070	RS	GPIO_EN1_CLR	GPIO Enable Bit 1 Atomic Clear Register
0x0074	R	RFU	Reserved
0x0078	R	RFU	Reserved
0x007C	R	RFU	Reserved
0x00B0	RW	GPIO_DS_SEL0	GPIO Drive Strength Selection Register 0
0x00B4	RW	GPIO_DS_SEL1	GPIO Drive Strength Selection Register 1
0x00B8	RW	GPIO_PSEL	GPIO Pulldown/Pullup Strength Register
0x00C0	R	RFU	Reserved

Refer to section 3.0 Digital Peripherals and System Control Registers for details on reset values and various access modes.

9.3.1 GPIO Function Enable Bit 0 Register (GPIO_EN0, Offset 0x0000)

Table 9-5. GPIO_EN0 (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	GPIO_EN0[31:24]							
Reset	1111 1111							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_EN0[23:16]							
Reset	1111 1111							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_EN0[15:8]							
Reset	1111 1111							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_EN0[7:0]							
Reset	1111 1111*							
Access	RW							

Name	Bits	Description	Settings
GPIO_EN0	31:0	This bit, in conjunction with bits in GPIO_EN1 configures the corresponding device pin for digital I/O or alternate functions.	This field can be modified directly by writing to this register or indirectly through GPIO_EN0_SET or GPIO_EN0_CLR. Input, interrupt, and PMU wake functions can always be used regardless of the setting of this bit.

9.3.2 GPIO Function Enable Bit 0 Set Register (GPIO_EN0_SET, Offset 0x0004)

Table 9-6. GPIO_EN0_SET (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	GPIO_EN0_SET[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_EN0_SET[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_EN0_SET[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_EN0_SET[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_EN0_SET	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_EN0 to '1', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_EN0 bit in this position to '1' Read: All reads return 0.

9.3.3 GPIO Function Enable Bit 0 Clear Register (GPIO_EN0_CLR, Offset 0x0008)

Table 9-7. GPIO_EN0_CLR (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	GPIO_EN0_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_EN0_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_EN0_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_EN0_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_EN0_CLR	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_EN0 to '0', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_EN0 bit in this position to '0' Read: All reads return 0.

9.3.4 GPIO Output Enable Register (GPIO_OUT_EN, Offset 0x000C)

Table 9-8. GPIO_OUT_EN (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_OUT_EN[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_OUT_EN[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_OUT_EN[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_OUT_EN[7:0]							
Reset	0000 0000*							
Access	RW							

Name	Bits	Description	Settings
GPIO_OUT_EN	31:0	Each bit controls the GPIO_OUT_EN setting for one GPIO pin in the associated port.	0: Disable GPIO output (default mode). The pad state is controlled by the GPIO_PDPUS_SEL0, GPIO_PDPUS_SEL1 and GPIO_PSSSEL bits). 1: Enable GPIO output. This bit setting has no effect when GPIO port is set to I/O function.

9.3.5 GPIO Output Enable Set Register (GPIO_OUT_EN_SET, Offset 0x0010)

Table 9-9. GPIO_OUT_EN_SET (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	GPIO_OUT_EN_SET[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_OUT_EN_SET[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_OUT_EN_SET[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_OUT_EN_SET[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_OUT_EN_SET	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_OUT_EN to '1', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_OUT_EN bit in this position to '1' Read: All reads return 0.

9.3.6 GPIO Output Enable Clear Register (GPIO_OUT_EN_CLR, Offset 0x0014)

Table 9-10. GPIO_OUT_EN_CLR (Offset 0x0014)

Position	31	30	29	28	27	26	25	24
Field	GPIO_OUT_EN_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_OUT_EN_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_OUT_EN_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_OUT_EN_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_OUT_EN_CLR	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_OUT_EN to '0', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_OUT_EN bit in this position to '0' Read: All reads return 0.

9.3.7 GPIO Output Register (GPIO_OUT, Offset 0x0018)

Table 9-11. GPIO_OUT (Offset 0x0018)

Position	31	30	29	28	27	26	25	24
Field	GPIO_OUT[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_OUT[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_OUT[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_OUT[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_OUT	31:0	Each bit controls the GPIO_OUT setting for one pin in the associated port. This register can be written either directly, or by using the GPIO_OUT_SET and GPIO_OUT_CLR registers.	0: Drive logic 0 (low) on GPIO output. 1: Drive logic 1 (high) on GPIO output This setting only takes effect when the GPIO_EN bit is set to 1 and GPIO_OUT_EN is set to 1 for the associated GPIO pin.

9.3.8 GPIO Output Set Register (GPIO_OUT_SET, Offset 0x001C)

Table 9-12. GPIO_OUT_SET (Offset 0x001C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_OUT_SET[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_OUT_SET[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_OUT_SET[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_OUT_SET[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_OUT_SET	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_OUT to '1', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_OUT bit in this position to '1' Read: All reads return 0.

9.3.9 GPIO Output Clear Register (GPIO_OUT_CLR, Offset 0x0020)

Table 9-13. GPIO_OUT_CLR (Offset 0x0020)

Position	31	30	29	28	27	26	25	24
Field	GPIO_OUT_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_OUT_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_OUT_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_OUT_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_OUT_CLR	31:0	Writing a 1 to one or more bits in this register clears the bits in the same positions in GPIO_OUT to '0', without affecting other bits in that register.	Write 0: No effect 1: Clear GPIO_OUT bit in this position to '0' Read: All reads return 0.

9.3.10 GPIO Input Register (GPIO_IN, Offset 0x0024)

Table 9-14. GPIO_IN (Offset 0x0024)

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

Name	Bits	Description	Settings
GPIO_IN	31:0	Read-only register to read from the logic states of the GPIO pins on this port.	Each bit in this register reads '0' or '1' depending on the logic state present on the associated GPIO pin. This register can be read regardless of the configuration/mode of the pin (GPIO input, GPIO output, alternate function, etc.)

9.3.11 GPIO Interrupt Mode Register (GPIO_INT_MOD, Offset 0x0028)

Table 9-15. GPIO_INT_MOD (Offset 0x0028)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_MOD[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_MOD[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_MOD[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_MOD[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_INT_MOD	31:0	Each bit in this register controls the interrupt mode setting for the associated GPIO pin on this port.	0: Interrupts for this pin are level triggered 1: Interrupts for this pin are edge triggered This setting has no effect unless GPIO_INT_EN is set to 1 for the associated pin(s).

9.3.12 GPIO Interrupt Polarity Register (GPIO_INT_POL, Offset 0x002C)

Table 9-16. GPIO_INT_POL (Offset 0x002C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_POL[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_POL[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_POL[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_POL[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_INT_POL	31:0	Each bit in this register controls the interrupt polarity setting for one GPIO pin in the associated port.	0: Interrupts are latched on a falling edge (if GPIO_INT_MOD=1) or low-level (GPIO_INT_MOD=0) condition for this pin. 1: Interrupts are latched on a rising edge (if GPIO_INT_MOD=1) or high-level (GPIO_INT_MOD=0) condition for this pin. This setting has no effect unless GPIO_INT_EN is set to 1 for the associated pin(s). If GPIO_INT_MOD=1 and GPIO_INT_DUAL_EDGE=1 for this pin, this setting has no effect since both rising and falling edges are used for triggering.

9.3.13 GPIO Input Enable Register (GPIO_IN_EN, Offset 0x0030)

Table 9-17. GPIO_IN_EN (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	GPIO_IN_EN[31:24]							
Reset	1111 1111							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_IN_EN[23:16]							
Reset	1111 1111							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_IN_EN[15:8]							
Reset	1111 1111							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_IN_EN[7:0]							
Reset	1111 1111							
Access	RW							

Name	Bits	Description	Settings
GPIO_IN_EN	31:0	Each bit in this register controls the GPIO input enable for the associated pin on the GPIO port.	0: Input is not connected 1: Input pin connected to the pad for reading via GPIO_IN Register.

9.3.14 GPIO Interrupt Enable Register (GPIO_INT_EN, Offset 0x0034)

Table 9-18. GPIO_INT_EN (Offset 0x0034)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_EN[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_EN[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_EN[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_EN[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_INT_EN	31:0	Each bit in this register controls the GPIO interrupt enable for the associated pin on the GPIO port.	0: Interrupts are disabled for this GPIO pin. 1: Interrupts are enabled for this GPIO pin. Clearing this bit to 0 do not clear pending interrupts (GPIO_INT_STAT). These must be cleared separately.

9.3.15 GPIO Interrupt Enable Set Register (GPIO_INT_EN_SET, Offset 0x0038)

Table 9-19. GPIO_INT_EN_SET (Offset 0x0038)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_EN_SET[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_EN_SET[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_EN_SET[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_EN_SET[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_INT_EN_SET	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_INT_EN to '1', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_INT_EN bit in this position to '1' Read: All reads return 0.

9.3.16 GPIO Interrupt Enable Clear Register (GPIO_INT_EN_CLR, Offset 0x003C)

Table 9-20. GPIO_INT_EN_CLR (Offset 0x003C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_EN_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_EN_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_EN_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_EN_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_INT_EN_CLR	31:0	Writing a 1 to one or more bits in this register clears the bits in the same positions in GPIO_INT_EN to '0', without affecting other bits in that register.	Write 0: No effect 1: Clear GPIO_INT_EN bit in this position to '0' Read: All reads return 0.

9.3.17 GPIO Interrupt Status Register (GPIO_INT_STAT, Offset 0x0040)

Table 9-21. GPIO_INT_STAT (Offset 0x0040)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_STAT[31:24]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_STAT[23:16]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_STAT[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_STAT[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
GPIO_INT_STAT	31:0	Each bit in this register contains the pending interrupt status for the associated GPIO pin in this port.	0: No interrupt is pending on this GPIO pin. 1: An interrupt is pending (and has not yet been serviced) on this GPIO pin. This register cannot be written directly. Write to GPIO_INT_CLR to clear bits in this register.

9.3.18 GPIO Interrupt Clear Register (GPIO_INT_CLR, Offset = 0x0048)

Table 9-22. GPIO_INT_CLR (Offset 0x0048)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_INT_CLR	31:0	Writing a 1 to one or more bits in this register clears the bits in the same positions in GPIO_INT_STAT to '0', without affecting other bits in that register.	Write 0: No effect 1: Clear GPIO_INT_STAT bit in this position to '0'. This clears the pending interrupt (if any) and resets the edge detectors for this pin. Read: All reads return 0.

9.3.19 GPIO Wake Enable Register (GPIO_WAKE_EN, Offset 0x004C)

Table 9-23. GPIO_WAKE_EN (Offset 0x004C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_WAKE_EN[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_WAKE_EN[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_WAKE_EN[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_WAKE_EN[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_WAKE_EN	31:0	Each bit in this register controls the PMU wakeup enable for the associated GPIO pin in this port.	0: PMU wakeup for this GPIO is disabled. 1: PMU wakeup for this GPIO is enabled. The PMU must be configured to accept GPIO as a wakeup source for this to have any effect.

9.3.20 GPIO Wake Enable Set Register (GPIO_WAKE_EN_SET, Offset 0x0050)

Table 9-24. GPIO_WAKE_EN_SET (Offset 0x0050)

Position	31	30	29	28	27	26	25	24
Field	GPIO_WAKE_EN_SET[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_WAKE_EN_SET[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_WAKE_EN_SET[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_WAKE_EN_SET[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_WAKE_EN_SET	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_WAKE_EN to '1', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_WAKE_EN bit in this position to '1' Read: All reads return 0.

9.3.21 GPIO Wake Enable Clear Register (GPIO_OUT_EN_CLR, Offset 0x0054)

Table 9-25. GPIO_WAKE_EN_CLEAR (Offset 0x0054)

Position	31	30	29	28	27	26	25	24
Field	GPIO_WAKE_EN_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_WAKE_EN_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_WAKE_EN_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_WAKE_EN_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_WAKE_EN_CLR	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_WAKE_EN to '0', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_WAKE_EN bit in this position to '0' Read: All reads return 0.

9.3.22 GPIO Interrupt Dual Edge Mode Register (GPIO_INT_DUAL_EDGE, Offset 0x005C)

Table 9-26. GPIO_INT_DUAL_EDGE (Offset 0x005C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_INT_DUAL_EDGE[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_INT_DUAL_EDGE[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_INT_DUAL_EDGE[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_INT_DUAL_EDGE[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_INT_DUAL_EDGE	31:0	Each bit in this register selects the dual-edge mode for the associated GPIO pin in this port.	0: No effect. 1: Dual-edge mode is enabled. If edge-triggered interrupts are enabled on this GPIO pin, then both the rising and falling edges trigger interrupts regardless of the GPIO_INT_POL setting.

9.3.23 GPIO Input Mode Selection Register 0 (GPIO_PDPU_SEL0, Offset 0x0060)

Table 9-27. GPIO_PDPU_SEL0 (Offset 0x0060)

Position	31	30	29	28	27	26	25	24
Field	GPIO_PDPU_SEL0[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_PDPU_SEL0[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_PDPU_SEL0[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_PDPU_SEL0[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	SEL1:SEL0	Effect
GPIO_PDPU_SEL0	31:0	The two bits in GPIO_PDPU_SEL0 and GPIO_PDPU_SEL1 for each GPIO pin work together to determine the pad mode when the GPIO is set to input mode. Please refer to Table 9-1 for complete description of the input set up configuration.	00	High impedance
GPIO_PDPU_SEL1	31:0		01	Pullup mode
			10	Pulldown mode
			11	[Reserved]

9.3.24 GPIO Input Mode Selection Register 1 (GPIO_PDPU_SEL1, Offset 0x0064)

Table 9-28. GPIO_PDPU_SEL1 (Offset 0x0064)

Position	31	30	29	28	27	26	25	24
Field	GPIO_PDPU_SEL1[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_PDPU_SEL1[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_PDPU_SEL1[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_PDPU_SEL1[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_PDPU_SEL1	31:0	See description for GPIO_PDPU_SEL0.	

9.3.25 GPIO Function Enable Bit 1 Register (GPIO_EN1, Offset 0x0068)

Table 9-29. GPIO_EN1 (Offset 0x0068)

Position	31	30	29	28	27	26	25	24
Field	GPIO_EN1[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	GPIO_EN1[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	GPIO_EN1[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	GPIO_EN1[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
GPIO_EN1	31:0	This bit, in conjunction with bits in GPIO_EN0, configures the corresponding device pin for digital I/O or alternate functions.	This field can be modified directly by writing to this register or indirectly through GPIO_EN1_SET or GPIO_EN1_CLR.

9.3.26 GPIO Function Enable Bit 1 Set Register (GPIO_EN1_SET, Offset 0x006C)

Table 9-30. GPIO_EN1_SET (Offset 0x006C)

Position	31	30	29	28	27	26	25	24
Field	GPIO_EN1_SET[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_EN1_SET[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_EN1_SET[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_EN1_SET[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_EN1_SET	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_EN1 to '1', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_EN1 bit in this position to '1' Read: All reads return 0.

9.3.27 GPIO Function Enable Bit 1 Clear Register (GPIO_EN1_CLR, Offset 0x0070)

Table 9-31. GPIO_EN1_CLR (Offset 0x0070)

Position	31	30	29	28	27	26	25	24
Field	GPIO_EN1_CLR[31:24]							
Reset	0000 0000							
Access	RS							
Position	23	22	21	20	19	18	17	16
Field	GPIO_EN1_CLR[23:16]							
Reset	0000 0000							
Access	RS							
Position	15	14	13	12	11	10	9	8
Field	GPIO_EN1_CLR[15:8]							
Reset	0000 0000							
Access	RS							
Position	7	6	5	4	3	2	1	0
Field	GPIO_EN1_CLR[7:0]							
Reset	0000 0000							
Access	RS							

Name	Bits	Description	Settings
GPIO_EN1_CLR	31:0	Writing a 1 to one or more bits in this register sets the bits in the same positions in GPIO_EN1 to '0', without affecting other bits in that register.	Write 0: No effect 1: Set GPIO_EN1 bit in this position to '0' Read: All reads return 0.

9.3.28 GPIO Drive Strength Register 0 (GPIO_DS_SEL0, Offset 0x00B0)

Table 9-32. GPIO_DS_SEL0 (Offset 0x00B0)

Position	31	30	29	28	27	26	25	24
Field	DS_SEL0[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DS_SEL0[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DS_SEL0[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DS_SEL0[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DS_SEL0	31:0	The two bits in GPIO_DS_SEL0 and GPIO_DS_SEL1 for each GPIO pin work together to determine the drive strength when the GPIO is set to the output mode. Please refer to Table 9-2 for complete description of the output setup of drive strength.	

9.3.29 GPIO Drive Strength Register 1 (GPIO_DS_SEL1, Offset 0x00B4)

Table 9-33. GPIO_DS_SEL1 (Offset 0x00B4)

Position	31	30	29	28	27	26	25	24
Field	DS_SEL1[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DS_SEL1[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DS_SEL1[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DS_SEL1[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DS_SEL1	31:0	Each bit in this register selects the drive strength for the associated GPIO pin in this port. Refer to the data sheet for sink/source current of GPIO pins in each mode.	0: GPIO port pin is in low-drive mode 1: GPIO port pin is in high-drive mode

9.3.30 GPIO Pulldown/Pullup Strength Register (GPIO_PSSEL, Offset 0x00B8)

Table 9-34. GPIO_PSSEL (Offset 0x00B8)

Position	31	30	29	28	27	26	25	24
Field	PSSEL[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	PSSEL[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	PSSEL[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	PSSEL[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
PSSEL	31:0	GPIO Pulldown/Pullup Strength Select. Selects the strength of the pullup or pulldown resistor for a pin configured for input mode. Please refer to Table 9-1 for complete description of the input set up configuration.	0: Weak pulldown/pullup resistor for input pin. 1: Strong pulldown/pullup resistor for input pin. Note: Refer to the data sheet for specific electrical characteristics of the Pulldown/Pullup resistances.

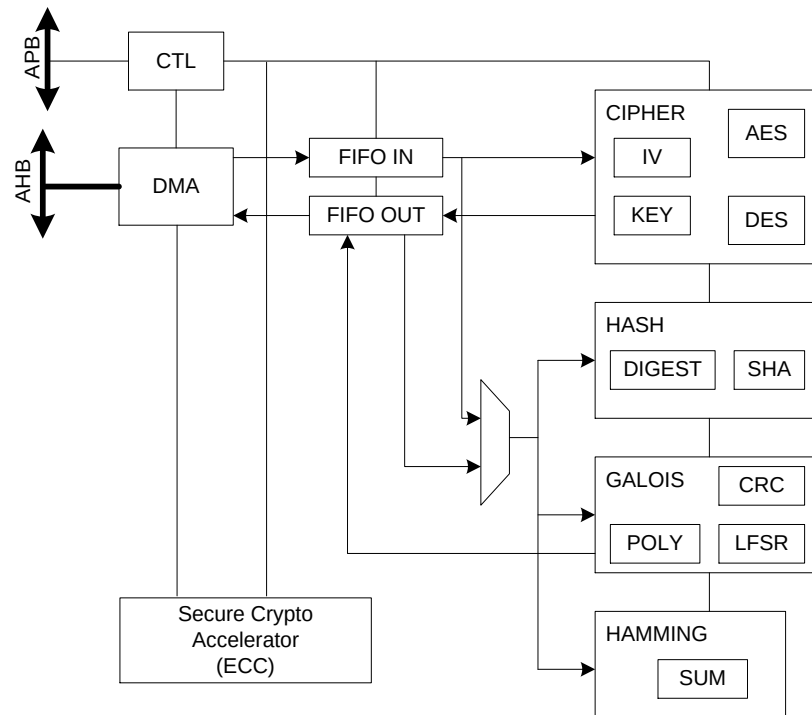
10.0 Cryptographic Accelerator

The cryptographic accelerator assists the computationally intensive operations of several common algorithms. The supported algorithms include:

- AES-128, 192, and 256 (FIPS 197)
- DES and 3DES (NIST SP800-67)
- SHA-1, 224, 256, 384, and 512 (FIPS 180-3)
- Programmable CRC generator
- Hamming code generator
- Support for NIST approved block modes (SP800-38A)
- Integrated DMA with read ahead and write buffers for high throughput
- Parallel calculation of blockcipher and hash functions
- Dedicated asymmetrical engine optimized for Elliptic Curve Cryptography (ECC)

The cryptographic accelerator is configured through an APB register interface.

Figure 10-1. Crypto Accelerator Block Diagram



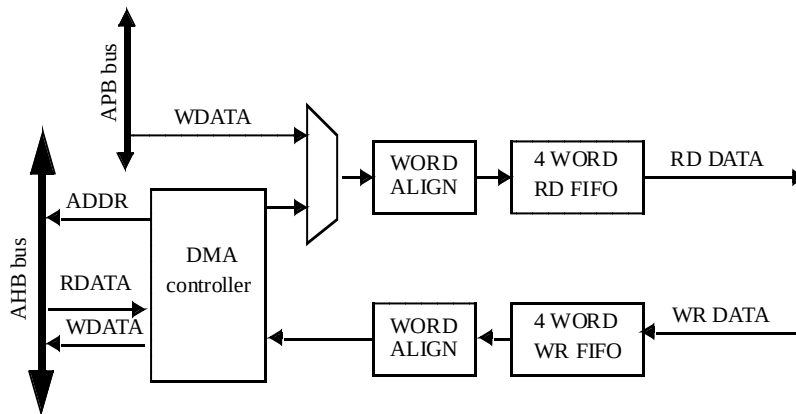
10.1 Crypto Buffers

10.1.1 Direct Memory Access

To efficiently encrypt or hash data, a DMA is integrated with the crypto accelerator. The DMA starts operating when the DMA count is written with a non-zero value. Data is read from the source address and written to the destination address. The address generators automatically increment. The source and destination of the DMA engine can point to the same memory location to encrypt or decrypt where the data is placed. The FIFOs have word alignment buffers to allow data to begin or end on byte boundaries. The DMA bursts data four words at a time except for the beginning or end of the data if it is unaligned.

While the crypto accelerator is busy encrypting or hashing data, the DMA will prefetch the data for the next operation and store it in the read FIFO. Once the cipher or hash generator is done, the data for the next operation is immediately available. Data output is buffered in the write FIFO so the next cipher or hash operation can immediately begin calculating on the next block. This keeps the cipher and hash generator running continuously without having to wait for data to be written to or read from the bus. The block cipher can operate in parallel with the hash accelerator if only one operation uses the DMA. The hash generator input can come from the input or output FIFO.

Figure 10-2. DMA Block Diagram



10.1.2 FIFOs

The read FIFO and write FIFO have programmable sources to allow flexibility in their operation. Data written to the write FIFO is always written out in the AHB DMA. The read and write FIFOs can come from the following sources.

Read FIFO

- APB
- AHB DMA
- Random Number Generator

Write FIFO

- None
- Cipher output
- Read FIFO

During crypto operation, a typical setup is to use the AHB DMA as the read FIFO source and the cipher output as the write FIFO source. This reads data from memory and writes the encrypted or decrypted result back out to memory.

A Cipher based Message Authentication Code (CMAC) is like a digital signature or a Keyed-Hash Message Authentication Code (HMAC). It uses a cipher in a block chaining mode to form a cryptographic checksum. In this

mode of operation, the AHB DMA is used as the read FIFO source, but the cipher output is not written back to memory. Only the final cipher block is of interest. So, the write FIFO source should be set to none.

The DMA can be used to copy memory like the memcpy standard C function by setting the write FIFO source to the read FIFO. If the Hamming ECC generator is enabled, Flash pages can be copied to memory while simultaneously calculating the error correction code.

Memory can be filled with a block of data like the memset() standard C function by pointing the write FIFO source to the read FIFO and setting the read FIFO to the APB. Similarly, memory can be filled with random data by pointing the read FIFO source to the random number generator and the write FIFO source to the read FIFO.

Deterministic Random Number Generators (DRNG) such as those specified in NIST SP 800-90 can be implemented by feeding a source of entropy into a hash function or cipher-based MAC. This can be done by setting the read FIFO source to the random number generator, disabling the write FIFO, and enabling the hash generator or cipher.

10.2 Symmetric Block Cipher Accelerator

The symmetric block cipher accelerator supports the Advanced Encryption Standard (AES) family of block ciphers (AES-128, AES-192, and AES-256), the Data Encryption Standard (DES), and Triple Data Encryption Algorithm (TDEA).

The symmetric block ciphers encrypt or decrypt data in blocks. The block sizes for each cipher is given in the table below.

Table 10-1. Symmetric Block Ciphers

Cipher	Key Size	Used Key Bits	Effective Strength*	Block Size
TDEA	192-bits	168-bits	112-bits	64-bits
AES-128	128-bits	128-bits	128-bits	128-bits
AES-192	192-bits	192-bits	192-bits	128-bits
AES-256	256-bits	256-bits	256-bits	128-bits

*NIST SP800-57

The block ciphers can be used in various modes of operation. In the simplest mode (ECB), each data block is simply encrypted or decrypted using the cipher. A side effect of this is that identical data blocks encrypt to the same ciphertext. Various modes of operation are used, which chain or feedback ciphertext from the previous block to seed the next encryption operation. This causes identical plaintext data blocks to encrypt to different ciphertexts. NIST SP800-38A specifies five modes of operation:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)

Also, SP800-38C/D recommends the following modes:

- Counter with CBC-MAC (CCM)
- Galois Counter Mode (GCM)

For the CFB mode of operation, the mode size is equal to the block size. 128-bit CFB is supported for AES and 64 bit CFB is supported for TDEA. 1-bit CFB and 8-bit CFB are not supported. For the CTR mode of operation, the lower 32-bits of the initial vector increment.

AES GCM only works on 96-bits initial vectors (IV).

The cipher algorithm and mode of operation are set in the cipher control register. The cipher starts operating once the FIFO is full.

10.2.1 AES GCM and CCM

When using AES GCM, each time the AES key is changed, software is responsible to compute a new H vector corresponding to the key by setting the CIPHER_CTRL.COMPH bit. This bit is automatically cleared by hardware when the computation starts. When the computation is completed, CRYPTO_CTRL.CPH_DONE is set.

Before starting an AES GCM or CCM computation, the AAD Length (A_LENGTH) and PLD Length (PLD_LENGTH) registers must be set. These two lengths are 64 bits wide. The initialization vector must be set in the CIPHER_INIT registers and CIPHER_CTRL.MODE must be set to GCM or CCM. For the CCM mode, CCM parameters M and L must also be set in CIPHER_CTRL.CCMM CIPHER_CTRL.CCML fields. The computation starts when data is loaded.

If existing, 128 bits AAD blocks must be loaded first (CIPHER_CTRL.DTYPE must be set to 0). Data must be left aligned and right padded with zeros if required. When AES is ready to receive a new AAD block, CIPHER_CTRL.RDY is set.

After AAD, if existing, payload blocks must be loaded (CIPHER_CTRL.DTYPE must be set to 1). Data must be left aligned and right padded with zeros if required. When loading PLD blocks, CIPHER_CTRL.RDY indicates that the corresponding cipher block is ready to be fetched and that a new block can be loaded (if needed).

When computation is completed, CRYPTO_CTRL.CPH_DONE is set and the authentication tag is available in the TAGMIC registers.

10.3 Hash Function Accelerator

The Cryptographic Hash Function accelerator supports SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 hash algorithms. It takes an input message of arbitrary length and summarizes it in a fixed length message digest.

Data is processed in 512-bit (16 word, for SHA-1, SHA-224 and SHA-256) or 1024-bit (32 word, for SHA-384 and SHA-512) blocks. After every 16 or 32 words are written to the hash block, several cycles are needed to complete the hash of that block. The number of cycles needed to compute the hash is dependent upon the number of rounds in the algorithm. The first 16 rounds are calculated as the 16 or 32 words are written to the hash block. The remaining rounds are calculated at a rate of one clock cycle per round.

Table 10-2. Hash Functions

Algorithm	Digest Length	Effective Strength*	Block Size
SHA-1	160 bits	63 bits	512 bits
SHA-224	224 bits	112 bits	512 bits
SHA-256	256 bits	128 bits	512 bits
SHA-384	384 bits	192 bits	1024 bits
SHA-512	512 bits	256 bits	1024 bits

* NIST SP800-57

The integrated DMA is capable of fetching data for the hash accelerator. The hash can be calculated in parallel with the block cipher if only one of the operations uses the DMA.

The software must manage the cache coherency. After the crypto block reset, the CRYPTO_CTRL.RDY must be polled in the software before any other actions can start. Since the crypto DMA does not interact with the CPU caches, prior to any DMA transfer, the caches storing the source and destination memory ranges must be 'cleaned'. Also, after the DMA transfer, caches storing the destination memory range must be 'invalidated'.

Once all data has been written to the CRYPTO Data register, the final message block must be padded according to the FIPS-180-2 standard. The standard requires that the bit length be appended to the end of the message. The length of the message in binary representation is contained in the HASH Message Size Registers (HASH_MSG_SZ). Prior to hashing the last message data, the Last Message Bit (HASH_CTRL.LAST) should be set along with the HASH_MSG_SZ registers. The HASH_MSG_SZ value is automatically padded to the last message block.

For SHA-1 and SHA-256, a single bit equal to '1' is appended to the end of the message. The message is then padded using software with zeros until 64 bits remain in the last 512-bit block. If less than 64 bits remain, then zeros are appended to the message until 64 bits remain in the next 512-bit block. The HASH_CTRL.LAST bit is set along with the HASH_MSG_SZ[1:0]. The hardware appends values in these registers to the last 64 bits of the final message block.

For SHA-384 and SHA-512, a single bit equal to '1' is appended to the end of the message. The message is then padded using software with zeros until 128 bits remain in the last 1024-bit block. If less than 128 bits remain, then zeros are appended to the message until 128 bits remain in the next 1024-bit block. The HASH_CTRL.LAST bit is set along with the HASH_MSG_SZ[3:0]. The hardware appends values in these registers to the last 128 bits of the final message block.

10.4 Galois Field Accelerator

The Galois (gal-wah) accelerator can be used to generate the Cyclic Redundancy Checks (CRC) of messages or to generate pseudo random numbers using a Linear Feedback Shift Register (LFSR). The Galois Fields are sometimes denoted $GF(2^n)$, where n denotes the highest order term of the reduction polynomial and 2^n denotes the size of the field.

10.4.1 Cyclic Redundancy Check

Cyclic Redundancy Checks (CRC) are commonly used for error detection. An n -bit CRC can detect the following types of errors.

- All single-bit errors.
- All two-bit errors for block lengths less than 2^k , where k is the order of the longest irreducible factor of the polynomial
- All odd number of errors for polynomials with the parity polynomial $(x+1)$ as one of its factors (polynomials with an even number of terms).
- All burst errors less than n bits.
- Overall, all except 1 out of 2^n errors are detected.
 1. 99.998% for a 16-bit CRC
 2. 99.99999998% for a 32-bit CRC

The crypto accelerator can calculate the CRC of a block of data. The data is written to the CRYPTO Data register.

In practice, the starting initial CRC value is typically preset to all ones. If the CRC was preset to all zeros and an initial stream of all zeros is received, the CRC does not change. Therefore, most implementations preset the CRC to all ones or some constant value.

Historically, CRCs were calculated on serial bit streams. Most serial bit streams were sent least significant bit first. The CRC was calculated as each bit was transmitted or received. This resulted in the CRC being calculated on the least significant bit of the data first.

The CRC is typically appended to the end of the data. If the receiver calculates the CRC on both the data and received CRC, the result should be all zeros if the data and CRC was received error free. Most implementations do not like to

check against an all zero checksum. Therefore, most implementations invert the CRC before transmitting it. By inverting the CRC on the transmitting end, the resulting CRC on the receiving end should be a constant. The specific constant is dependent upon the CRC polynomial. This works because the non-inverted CRC calculated at the end of the data XOR'd with the received inverted CRC is all ones ($CRC \oplus \sim CRC = 1s$). Shifting all ones through the polynomial results in the same constant for each message. The constant is dependent upon the polynomial.

Since the receiving end calculates a new CRC on both the data and received CRC, the received CRC must be sent in the correct order so the highest order term of the CRC is shifted through the generator first. Since data is typically shifted through the generator LSB first, the CRC is reversed bit-wise so the highest order term of the remainder is in the LSB position. Software CRC algorithms typically handle this by calculating everything backwards. They reverse the polynomial and do right shifts on the data. The resulting CRC ends up being bit swapped and in the correct format.

The CRYPTO_CRC register is preset to all ones if the crypto block is reset. The initial CRC state can be written to any value. The final inversion must be done by the software, if required.

The CRC generator has a programmable polynomial up to 32-bits. The polynomial should be written to the CRC_POLY register. The largest term x^n defines the length of the CRC. When calculating the CRC on data LSB first, the polynomial should be reversed so that the coefficient of the highest power term is in the LSB position. The largest term x^n is implied (always one) and should be omitted when writing to the CRC_POLY register. This is necessary because the polynomial is always one bit larger than the resulting CRC. So, a 32-bit CRC has a polynomial with 33 terms (x^0 to x^{32}).

CRC polynomials with good error detection properties should be irreducible (the polynomial should not be factorable). Therefore, the constant term x^0 or 1 should always be present, otherwise the polynomial would be factorable by x . If the constant term x^0 or 1 were not present, the resulting CRC would be cyclic with a subgroup smaller than x^n . The effective length of the CRC would be the difference between the highest and lowest order terms. Therefore, the highest and lowest order terms x^n and x^0 should always appear in the polynomial.

When found in literature, sometimes the LSB or MSB of the polynomial is omitted when the polynomial is written in binary. It is more common to see CRC polynomials with the MSB implied because that is the shifted off bit, XOR'd with the data, and tested to see if the result is set. Some literature assumes the reader knows that an n -bit CRC must have the x^n term set, or else it would be a smaller length CRC.

Some common CRC polynomials and their check constants are shown in the table below. The polynomial register resets to the 32-bit CRC polynomial used by Ethernet, PPP, and file compression utilities such as zip or gzip.

Table 10-3. Common CRC Polynomials

Algorithm	Polynomial Expression	Order	Polynomial	Check
CRC-32 Ethernet	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$	LSB	0xEDB88320	0xDEBB20E3
CRC-CCITT	$x^{16} + x^{12} + x^5 + x^0$	LSB	0x00008408	0x0000F0B8
CRC-16	$x^{16} + x^{15} + x^2 + x^0$	LSB	0x0000A001	0x0000B001
USB data	$x^{16} + x^{15} + x^2 + x^0$	MSB	0x80050000	0x800D0000
Parity	$x^1 + x^0$	LSB	0x00000001	

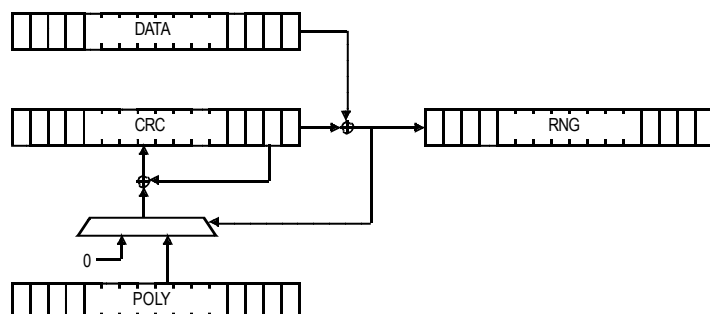
By default, the CRC accelerator does right shifts and calculates the CRC on the LSB of the data first. The CRC can be calculated on the MSB of the data first by setting the bit-swap control bit. To calculate the CRC MSB first, you must left justify the polynomial in the CRC_POLY register. The hardware implies the MSB of the polynomial just as it did when shifting the LSB first. The LSB of the polynomial should be set. This defines the length of the CRC. The

initial state of the CRC should also be left justified. When the CRC calculation is complete, it is necessary to right shift the CRC to right justify it if the polynomial is less than 32-bits.

10.4.2 Linear Feedback Shift Register

Linear Feedback Shift Registers (LFSR) are commonly used to implement Pseudo Random Number Generators (PRNG). A LFSR polynomial can be written to the CRC_POLY register to generate pseudo-random data. The starting state or seed for the pseudo random sequence should be written to the CRC register. The lockup state of all zeros is detected and the LFSR substitutes the value 1 to prevent lockup.

Figure 10-3. Galois Field CRC and LFSR Architecture



Different polynomials generate different sequences of random data. Ideally, an n-bit polynomial generates a random sequence of $2^n - 1$ bits. Not all polynomials are maximal length. Some repeat before the theoretical maximum length of $2^n - 1$. There are thousands of different maximal length 32-bit LFSR polynomials. Any length of an LFSR polynomial can be used up to 32-bits. Some tables of maximal length LFSR polynomials omit the MSB x^n term or the LSB $x^0 = 1$ terms. Fibonacci LFSRs feedback the XOR of all the taps to the constant term $x^0 = 1$. So, it is often implied when listing the taps, but must be present when writing the polynomial to the CRC_POLY register.

The crypto accelerator automatically generates the next sequence of 32-bits whenever the CRYPTO_LFSR register is read. If the PRNG control bit is set, the incoming data will be forced to zero. The DMA can be used to quickly fill a block of memory with pseudo-random data.

10.4.3 Entropy

The crypto block has a high-frequency ring oscillator (approximately 1GHz), which is sampled every system clock to generate entropy.

The raw output of the ring oscillator can be observed to determine how much entropy it provides. Given the probability of predicting the ring oscillator output, the amount of entropy provided per bit can be calculated as:

$$\text{entropy} = -\log_2(\text{predictability})$$

If the output can be predicted with 99% certainty, then 0.0145 bits of entropy are provided per bit of output. To obtain 32 bits of entropy, a minimum of 2207 bits of output from the ring oscillator must be mixed. If the probability of predicting the output is 50%, 1 bit of entropy is provided per bit of output.

10.5 Hamming Code Generator

The Hamming code generator can be used to calculate an Error Correction Code (ECC) on a block of data. One application of Hamming codes is for Single Level Cell (SLC) NAND Flash memories. Multi-Level Cell (MLC) Flash

memories require Error Correction Codes that can correct multiple bit errors since a corrupt cell can have multiple bit errors.

Hamming codes are capable of correcting single bit errors. An extra parity bit can be included to detect two-bit errors. This is commonly referred to as Signal Error Correction, Double Error Detection (SECDED). Three errors masquerade as a correctable single bit error.

The Hamming code generator calculates the ECC on a block of data up to 2^{16} bits in length (8k bytes). Software implementations can extend this to any length. Since the Hamming code can only correct a single bit error, increasing the block size increases the likelihood of multiple errors which are uncorrectable. The Hamming code generator in the crypto accelerator generates even parity on even halves of bit groups.

If the user wants the parity of the odd halves of the bit groups, they can XOR the parity of the even halves with the parity of the entire array. If the parity of the entire array is odd, the parity of the odd halves is the inverse of the parity of the even halves. If the parity of the entire array is even, the parity of the odd halves is identical to the parity of the even halves.

Figure 10-4. Hamming XOR Calculation

b7	b6	b5	b4	b3	b2	b1	b0	byte 5
b7	b6	b5	b4	b3	b2	b1	b0	byte 4
b7	b6	b5	b4	b3	b2	b1	b0	byte 3
b7	b6	b5	b4	b3	b2	b1	b0	byte 2
b7	b6	b5	b4	b3	b2	b1	b0	byte 1
b7	b6	b5	b4	b3	b2	b1	b0	byte 0
ECC bit0 – xor every other bit								
b7	b6	b5	b4	b3	b2	b1	b0	byte 5
b7	b6	b5	b4	b3	b2	b1	b0	byte 4
b7	b6	b5	b4	b3	b2	b1	b0	byte 3
b7	b6	b5	b4	b3	b2	b1	b0	byte 2
b7	b6	b5	b4	b3	b2	b1	b0	byte 1
b7	b6	b5	b4	b3	b2	b1	b0	byte 0
ECC bit2 – xor every other 4 bits								
b7	b6	b5	b4	b3	b2	b1	b0	byte 5
b7	b6	b5	b4	b3	b2	b1	b0	byte 4
b7	b6	b5	b4	b3	b2	b1	b0	byte 3
b7	b6	b5	b4	b3	b2	b1	b0	byte 2
b7	b6	b5	b4	b3	b2	b1	b0	byte 1
b7	b6	b5	b4	b3	b2	b1	b0	byte 0
ECC bit4 – xor every other 16 bits								
b7	b6	b5	b4	b3	b2	b1	b0	byte 5
b7	b6	b5	b4	b3	b2	b1	b0	byte 4
b7	b6	b5	b4	b3	b2	b1	b0	byte 3
b7	b6	b5	b4	b3	b2	b1	b0	byte 2
b7	b6	b5	b4	b3	b2	b1	b0	byte 1
b7	b6	b5	b4	b3	b2	b1	b0	byte 0
ECC bit5 – xor every other 32 bits								
b7	b6	b5	b4	b3	b2	b1	b0	byte 5
b7	b6	b5	b4	b3	b2	b1	b0	byte 4
b7	b6	b5	b4	b3	b2	b1	b0	byte 3
b7	b6	b5	b4	b3	b2	b1	b0	byte 2
b7	b6	b5	b4	b3	b2	b1	b0	byte 1
b7	b6	b5	b4	b3	b2	b1	b0	byte 0

For a block of data 2^n bits in length, $n+1$ ECC bits are needed for single bit error correction (ECC B0 through Bn = $n+1$ ECC bits). The ECC bit n indicates parity of the entire array. If it is different than the stored ECC bit n , it indicates an error in the data array. The location of the error can be determined using the lower n ECC bits (B0 to Bn-1).

To determine if an error occurred, the user should XOR the saved and calculated ECC bits. If the result is zero, no error occurred. If the result of the ECC XOR operation is not zero, the location of the failing bit is given by the inverse of the ECC XOR result. If the failing bit location is greater than or equal to the size of the data block (bit n of the inverted ECC XOR is set), then the error is in the ECC bits. The ECC bit that is corrupt is the bit that is set because of the XOR of the two sets of ECC bits.

$$\text{failing\_bit\_location} = \sim(\text{ecc\_saved} \wedge \text{ecc\_calculated})$$

An error in the most significant ECC bit n masquerades as an error in the most significant bit of the data. To properly detect and correct an error in the MSB of the ECC, the user can include an extra ECC bit ($n+2$ ECC bits). These two most significant ECC bits should be identical if they are error free. If they are different, the failing ECC bit can be determined by XORing the saved and calculated ECC bits.

This extra ECC bit can be saved at the cost of an additional parity calculation. If the MSB of the data is set, the ECC parity bits should be XOR'd with $2^{n+1}-1$ (n 1s). This effectively swaps the MSB of the data with the MSB of the ECC for purposes of parity calculation. When examining the result, if the ECC calculation indicates the MSB of the data is

corrupt (the result is $2^{n+1}-1$), the error is really in the MSB of the ECC. If the result indicates the error is in the MSB of the ECC (the result is 2^{n+1}), the error is really in the MSB of the data.

ECC bits n and above are all identical. They all indicate the parity of the entire array. To use a block size of 64k bits (8k bytes), the parity bit of the entire array (bit 16) can be duplicated to obtain the higher order ECC bits.

Hamming codes can be generated over larger blocks using software. After the Hamming code is generated for an 8k block of data, the software should examine the parity of the block of data just calculated (bit 16 of the ECC register). If the parity of the block is odd (parity bit is set), the software should XOR additional software maintained ECC bits with the inverse of the block address. If the parity of the block is even (parity bit of the block is clear), the software does not need to modify the additional ECC bits. The parity of the block (bit 16 of the ECC register) should be reset for each block, but the remaining ECC bits (B0-B15) should remain unchanged and accumulate their respective XORs throughout the entire array.

To achieve Double Error Detection (but not correction), another ECC parity bit can be included. If the result of the ECC XOR is not zero, then this extra ECC bit should also be set indicating an error has occurred. If this bit is clear after the ECC XOR operation, it means an even number of errors has occurred and the data is not correctable. This does not mean this extra ECC bit detects all even number of bit errors. It just indicates an even number of errors has occurred. It is possible for an even number of bit errors to masquerade as valid data. This extra ECC bit can detect two-bit errors. An odd number of bit errors always masquerade as a single correctable bit error. This is a limitation of using parity to indicate error. Parity can only detect an odd number of errors. For stronger error detection, a Cyclic Redundancy Check (CRC) can be used.

The lower n ECC bits (B0 to B $n-1$) are all that is needed to determine the location of a single bit error in the data. The next ECC bit (which is the parity of the entire array) is simply used to indicate there is an error in the data that needs to be corrected. Subsequent ECC bits are only needed to detect an error in this added ECC bit or to detect double bit errors. If an alternative error detection scheme is used, such as a CRC, then only n ECC bits are needed to find the location of a single bit error in the data array.

Since all CRCs with at least two terms in the generator polynomial detect all single bit errors, the ECC bit n , which is used to determine if there is a single bit error in the array by checking parity is unnecessary. A CRC polynomial with an even number of terms has the parity polynomial $(x+1)$ as one of its factors and checks parity as well. A CRC of sufficient length is also capable of detecting all two-bit errors, making an ECC bit added for double error detection unnecessary as well.

Since a single bit error in the CRC would be catastrophic, the CRC should also be protected with ECC bits. If the CRC is appended to the data for the Hamming code calculation, it can be protected with the same set of ECC bits used to protect the data by including one additional ECC bit to account for the increase in block size.

Most manufacturers of NAND Flash memories have application notes that describe the calculation of a variation of Hamming codes. These application notes calculate parity on both the odd and even halves of the bit groups. Thus, they require $2*n$ ECC bits, almost twice as many, as necessary. If the even half is XOR'd with the odd half, the result is the parity of the entire array. Instead of storing parity for both the even halves and odd halves of all bit groups (which requires $2*n$ ECC bits), an implementation only really needs to store the parity of the entire array ($n+1$ ECC bits). The odd half of ECC bits can be determined by XORing the even half of ECC bits with the parity of the entire array. The parity of the entire array is just the next bit in the ECC register. So, saving $n+1$ bits from this register inherently saves the parity of the entire array. Storing both the odd and even parity halves offer no more error protection than the methods described above. For both methods, an odd number of errors masquerade as a correctable single bit error and all double bit errors are detected.

To calculate the ECC of a new block of data, the RST bit of the CRC Control register should be set to reset the Hamming parity bits to zero and to also reset the internal byte counter.

10.6 Secure Crypto Accelerator (SCA)

This accelerator is dedicated to non-symmetrical crypto operations. It is optimized for ECC (Elliptic Curve Cryptography) operations. This block does not support high-level RSA operations. However, the large modular multiplication can be used by the software to accelerate RSA computations.

Jacobi coordinates are supported by the SCA IP to improve security level. Different Jacobi coordinates can represent the same point: $(X ; Y ; Z)$ and $(X*\lambda^2 ; Y*\lambda^3 ; Z*\lambda)$ are equivalent. Therefore, the SCA IP converts affine coordinates to a random Jacobian representation by using a secret random λ . This countermeasure aims to counter DPA/CPA (also EM equivalent) and template/comparative attacks.

The accelerator can be used to both perform high-level operation, such as ECDSA signature and verification (P-256 only), and low-level operations such as modular addition, subtraction, multiplication, division, and scalar operations. It can also support up to 521-bit operations for low-level operations.

Internal state machines are implemented to perform high-level operations, in this case the software is not solicited, instead for low-level operations the software has a direct access to the primary operations to implement high-level algorithm (other ECC, RSA...).

High-level and low-level operations are selected through dedicated opcodes. Operands and Results reside in system memory like SRAM, Flash, and APB registers are used to store the data pointers to system memory. Therefore, the SCA does not require dedicated memory.

This accelerator has the following features:

- ECC basic operations
- High-level and low-level operations
- APB interface for IP configuration
- AHB Master interface to access system memory (operands, results...)
- No need for dedicated memory
- Pointer based architecture to easily access SCA memory without memory transfer overhead.
- Interrupt management
- Jacobi and affine coordinates are supported
- DRS support (wipe SCA memory)
- SPA/DPA and fault attack countermeasures
- Pre-calculated point support to speed-up ECC processing (ECDSA P256)
- Custom curve parameters (Brainpool...)
- EdDSA algorithms like Ed25519 and Ed448 can also be implemented (supported in TLS 1.3), as well as X25519
- Dedicated input for Secure Private key

10.6.1 Basic Operations

The SCA uses system memory to read/store operands, working buffer, and results. The data pointers shall be stored in the SCA configuration registers before starting any operations.

The main purpose of the SCA IP is to perform ECDSA P256 signature and verification. Some P256 parameters are hardcoded in the design to simplify and speed-up these two operations. However, the SCA can be used in manual mode to perform other operations to cover a wide range of ECC (elliptic curves) based operation, like ECDH, various curves (Brainpool...), different curve sizes (384, 521) and other ECDSA algorithm (basic modular operations and ECC point operation are provided).

SCA_CN.MANPARAM is used to select P256 operations or to use manual parameters for the other curves. If SCA_CN.MANPARAM = 0, hardcoded parameters are used to deal with NIST P256 operations, otherwise if SCA_CN.MANPARAM = 1, the curve parameters must be set in the operation buffer (SCA_OP_BUFF_ADDR).

Manual Parameters

SCA_CN.ECCSIZE is used to select the maximum size of the curve to optimize the computation when manual parameters are used, for example to use a 224-bit curve, ECCSIZE must be 256-bits and to use a 400-bit curve, ECCSIZE must be 521-bit.

Note that large size can be used for small curve operation, but the size of the needed system memory and processing time is not optimized. This means that ECCSIZE set to 521 bits can be used to process 224-bit, 256-bit, or 384-bit curves and in this case unused bits (MSB) of the results must be ignored.

SCA_CN.ECCSIZE bits are only available when SCA_CN.MANPARAM is set, otherwise SCA_CN.ECCSIZE is forced to '01' (256-bit) by hardware to deal with the NIST P256 curve.

Before starting any SCA operation, set data pointers in the dedicated SCA registers, . The SCA engine uses these data pointers to read operands, store results, and locate the working buffer.

The data pointers must be stored in the following registers:

SCA_PPX_ADDR, SCA_PPY_ADDR, SCA_PPZ_ADDR, SCA_PQX_ADDR, SCA_PQY_ADDR,
SCA_PQZ_ADDR, SCA_RDSA_ADDR, SCA_RES_ADDR and SCA_OP_BUFF_ADDR

Each operation has specific needs in term of working memory. Therefore, the system memory allocation can be optimized for each operation. Refer to 10.6.3 Memory allocation for SCA data in system memory to allocate system memory resources.

Once all the data pointers have been set, and operands available in the right location in the system memory, the operation can be started by setting SCA_CN.STC bit.

10.6.1.1 SCA Modular Operations

These operations can be used to add, subtract, multiply, and divide big numbers. The maximum size of the operand is defined by SCA_CN.ECCSIZE when SCA_CN.MANPARAM = 1. Otherwise, a 256-bit operation is selected.

The modulo must be set by using SCA_CN.MONADDR and SCA_MOD.MODDATA registers.

Examples:

For a 10-bit modular operation:

MODULO = 0x32F

SCA_CN.MODADDR = 0

SCA_MOD.MODDATA = 0x0000032F (LSB)

Note: Unused bits (11 to 31) must be set to '0'

For a 256-bit modular operation

MODULO = 0xFC632551_F3B9CAC2_A7179E84_BCE6FAAD_FFFFFFFF_FFFFFFFF_00000000_FFFFFFFF

The modulo must be loaded in the SCA internal register:

```
SCA_CN.MODADDR = 0
SCA_MOD.MODDATA = 0xFFFFFFFF (LSB)
SCA_CN.MODADDR = 1
SCA_MOD.MODDATA = 0x00000000
SCA_CN.MODADDR = 2
SCA_MOD.MODDATA = 0xFFFFFFFF
SCA_CN.MODADDR = 3
SCA_MOD.MODDATA = 0xFFFFFFFF
SCA_CN.MODADDR = 4
SCA_MOD.MODDATA = 0xBCE6FAAD
SCA_CN.MODADDR = 5
SCA_MOD.MODDATA = 0xA7179E84
SCA_CN.MODADDR = 6
SCA_MOD.MODDATA = 0xF3B9CAC2
SCA_CN.MODADDR = 7
SCA_MOD.MODDATA = 0xFC632551 (MSB)
```

For a 521-bit modular operation, the MSB must be loaded at 0x16:

```
SCA_CN.MODADDR = 0
SCA_MOD.MODDATA = MOD0 (LSB)
SCA_CN.MODADDR = 1
SCA_MOD.MODDATA = MOD1
...
SCA_CN.MODADDR = 16
SCA_MOD.MODDATA = MOD16 (MSB)
Note: unused bits (522 to 543) must be set to '0'
```

Table 10-4. SCA Opcodes for Modular Operations

Operation	Opcode	module	Comment	Parameters ⁽¹⁾
MM: Modular Multiplication	0	MODOP	Calculation of a product of two long integers modulo n <i>Inputs: PPx, PPy and MOD</i> <i>Output: RESx</i> $RESx = (PPx * PPy) \text{ modulo } MOD$ <i>With $PPx < MOD$ and $PPy < MOD$</i>	PPX PPy RESX MOD
MD: Modular Division	1	MODOP	Calculation of a ration of two long integers modulo n <i>Inputs: PPx, PPy and MOD bit</i> <i>Output: RESx</i> $RESx = (PPx / PPy) \text{ modulo } MOD$ <i>With $PPx < MOD$ and $PPy < MOD$</i>	PPX PPy RESX MOD OPBUFF ⁽²⁾

Operation	Opcode	module	Comment	Parameters ⁽¹⁾
			Warning: MOD must be primary, with non-primary modulo the result is not guaranteed since the invert may not exist.	
MA: Modular Addition	2	MODOP	Calculation of a sum of two long integers modulo n <i>Inputs: PPx, PPy and MOD bit</i> <i>Output: RESx</i> $RESx = (PPx + PPy) \text{ modulo } MOD$ <i>With $PPx < MOD$ and $PPy < MOD$</i>	PPX PPY RESX MOD
MS: Modular Subtraction	3	MODOP	Calculation of a difference of two long integers modulo n <i>Inputs: PPx, PPy and MOD bit</i> <i>Output: RESx</i> $RESx = (PPx - PPy) \text{ modulo } MOD$ <i>With $PPx < MOD$ and $PPy < MOD$</i>	PPX PPY RESX MOD

10.6.1.2 SCA Opcodes for NIST P256

Table 10-5 describes the operations that can be used over the NIST P256 curve. The accelerator is optimized for P256. P256 curve parameters (including modulo and pre-computed points) are hardcoded. Therefore, the use of P256 operation is straight forward and user friendly.

For other curve types or sizes, the manual parameter mode (SCA_CN.MANPARAM=1) must be used to load curve parameters in OPBUFF before start of the operation.

Table 10-5. SCA Opcodes for NIST P256

Operation	Opcode	module	Comment	Parameters ⁽¹⁾
ECJAADD: ECC point operation, Jacobian + Affine	5	POP	ECC point addition, one is in Jacobi coordinates and the other one is in Affine coordinates. The result is in Jacobi coordinates. <i>Inputs: $PP(X,Y,Z)$ = Jacobi Point</i> <i>$PQ(x,y)$ = Affine Point</i> <i>MOD</i> <i>Output: $RES(X,Y,Z)$</i> $RES(X,Y,Z) = PP(X,Y,Z) + PQ(x,y) \text{ modulo } MOD$	PPX, PPY, PPZ PQX, PQY OPBUFF ⁽²⁾ RESX, RESY, RESZ
ECJASUB: ECC point operation, Jacobian - Affine	6	POP	ECC point subtraction, one is in Jacobi coordinates and the other one is in Affine coordinates. The result is in Jacobi coordinates. <i>Inputs: $PP(X,Y,Z)$ = Jacobi Point</i>	PPX, PPY, PPZ PQX, PQY OPBUFF ⁽²⁾

Operation	Opcode	module	Comment	Parameters ⁽¹⁾
			$PQ(x,y) = \text{Affine Point}$ MOD Output: $RES(X,Y,Z)$ $RES(X,Y,Z) = PP(X,Y,Z) - PQ(x,y) \text{ modulo } MOD$	RESX, RESY, RESZ
ECJDBL : ECC point operation, Jacobian double	7	POP	ECC point double in Jacobi coordinates Input: $PP(X,Y,Z)$ and MOD Output: $RES(X,Y,Z)$ $RES(X,Y,Z) = 2*PP(X,Y,Z) \text{ modulo } MOD$	PPX, PPY, PPZ OPBUFF ⁽²⁾ RESX, RESY, RESZ
ECJACTOAF : ECC point operation, coordinates conversion	10	POP	EEC Point coordinates conversion, from Jacobi to Affine Inputs: $PP(X,Y,Z)$ and MOD Output: $RES(X,Y)$ $RESx = PP(X)/PP(Z)^2 \text{ modulo } MOD$ $RESy = PP(Y)/PP(Z)^3 \text{ modulo } MOD$	PPX, PPY, PPZ OPBUFF ⁽²⁾ RESX, RESY
ECAFFTOJAC : ECC point operation, coordinates conversion	11	POP	EEC Point coordinates conversion, from Affine to Jacobi Inputs: $PP(x,y)$ and MOD Output: $RES(X,Y,Z)$ $RESx = PP(x)*RND^2 \text{ modulo } MOD$ $RESy = PP(y)*RND^3 \text{ modulo } MOD$ $RESz = RND \text{ modulo } MOD$	PPX, PPY OPBUFF ⁽²⁾ RESX, RESY, RESZ
ECPVERIF : EC point verification	12	POP	Simply verify if a given point P indeed belongs to the curve P256 Inputs: $PP(X,Y,Z)$ ⁽¹⁾ and MOD Output: PVF1: 0 ok, 1 if not on the curve ⁽¹⁾ For affine $PP(Z)$ must be set to 1	PPX, PPY, PPZ OPBUFF ⁽²⁾ RESX, RESY, RESZ
ECSPSCAL : EC secure scalar for P256 Q=kP	13	SCALOP	EC secure scalar that uses pre-calculated points from SCA memory (PC0 ... PCn) k must be stored in PPz Inputs: $PP(z)$ and MOD Output: $AFFJAC=0 \Rightarrow RES(x,y)$ Affine coordinates $RES(x,y) = PPz * PP(x,y)$ $RES(z)$ by SCA as working variable	PPZ OPBUFF ⁽²⁾ RESX, RESY RESZ
ECSCALQ : EC secure scalar for Diffie Hellman operation (kQ) With authenticated Q	14	ECDSA	EC secure scalar that uses point Q from SCA memory ($PQ(x,y)$) k must be stored in PPx Q must be stored in $PQ(x,y)$ Inputs: $PP(x)$, $PQ(x,y)$ and MOD Output: $RES(x,y)$ Affine coordinates	PPX, PPY, PPZ PQX, PQY, PQZ OPBUFF ⁽²⁾ RESX, RESY

Operation	Opcode	module	Comment	Parameters ⁽¹⁾
			$RES(x,y) = PPz * PP(x,y)$ $RES(z)$ by SCA as working variable	RESZ
ECDSASIG: ECDSA signature	15	ECDSA	ECDSA signature k (nonce) must be stored in PPz d (private key) must be stored in PQx H(m) must be stored in PQz <i>Inputs: PP(z), PQ(x,z) and MOD</i> <i>Output:</i> $RESy = r = kP(x)$ $RESx = s = k^{-1}(H(m) + d \cdot r)$ $RES(z)$ is used by SCA as working variable	PPX, PPY PPZ PQX, PQY PQZ OPBUFF ⁽²⁾ RESX, RESY RESZ
ECDSAVER: ECDSA verification	16	ECDSA	ECDSA verification r must be stored in RDSA s must be stored in PPx H(m) must be stored in PQz Q must be stored in PQ(x,y) <i>Inputs: RDSA, PP(x), PQ(x,y,z) and MOD</i> <i>Output:</i> If pass, RESx =0, PVF=0 else RESx ≠0, PVF =1. $RES(x,y,z)$ are used by SCA as working variables	RDSA PPX, PPY, PPZ PQX, PQY, PQZ OPBUFF ⁽²⁾ RESX, RESY, RESZ
ECSPC2kPA2vQ: EC unsecure 2kP + 2vQ for P256	18	SCALOP	Used for ECDSA verification P+Q must be stored in affine in PP P-Q must be stored in affine in PQ k MSBs must be stored in PQz MSBs k LSBs must be stored in PPz MSBs v MSBs must be stored in PQz LSBs v LSBs must be stored in PPz LSBs <i>Inputs: PP(x,y,z), PQ(x,y,z) and MOD</i> <i>Output:</i> $RES(x,y)$ Affine coordinates	PPX, PPY, PPZ PQX, PQY, PQZ OPBUFF ⁽²⁾ RESX, RESY RESZ

Operation	Opcode	module	Comment	Parameters ⁽¹⁾
			$RES(x,y) = (PQz \text{ MSBs} PPz \text{ MSBs}) * PP(x,y) + (PQz \text{ LSBs} PPz \text{ LSBs}) * PQ(x,y)$ <i>RES(z) by SCA as working variable</i>	
ECPSQ: EC P-Q, with hard coded P ⁽¹⁾ and result in affine Hard coded if SCACN.MANPARAM is not set, otherwise P is from OPBUFF (refer to manual parameter section 10.6.2)	19	SCALOP	Q must be stored in $PQ(x,y)$ <i>Inputs: $PQ(x,y)$ and MOD</i> <i>Output:</i> <i>$RES(x,y)$ Affine coordinates</i> $RES(x,y) = P - PQ$ <i>RES(z) by SCA as working variable</i>	PQX, PQY OPBUFF RESX, RESY RESZ
ECPAQ: EC P+Q, with hard coded P and result in affine Hard coded if SCACN.MANPARAM is not set, otherwise P is from OPBUFF (refer to Manual Parameters 10.6.2)	20	SCALOP	Q must be stored in $PQ(x,y)$ <i>Inputs: $PQ(x,y)$ and MOD</i> <i>Output:</i> <i>$RES(x,y)$ Affine coordinates</i> $RES(x,y) = P + PQ$ <i>RES(z) by SCA as working variable</i>	PQX, PQY OPBUFF RESX, RESY RESZ

(1) Corresponding APB registers must be filled with data pointers. Refer to 10.6.3 Memory allocation for SCA data in system memory.

(2) Operation buffer is needed to store/load intermediate results

10.6.2 Manual Parameters

In Manual parameter mode the Operation buffer (**OPBUFF**) must be filled with ECC parameters. The **base address** of the operation buffer is defined by the Operational Buffer Address Register in Table 10-37. The size of the operating buffer depends on the operand size (refer to SCA_CN.ECCSIZE). Note that the speed of the operations increase with the operand size.

RA, **RB**, **RC**, and **RD** are internal variables used by the SCA IP to store/load intermediate results.

The following tables describe the OPBUFF mapping to store ECC curve parameters.

10.6.2.1 Up to 256-bit operations (ECCSIZE = 0 or 1)

Table 10-6 Operation Buffer Mapping (up to 256-bit operands)

Offset	Parameter	Size (bits)	Description
0x000	Mod P	256	P modulo
0x020	Mod N	256	N modulo
0x040	CA	256	ECC – CA parameter
0x060	CB	256	ECC – CB parameter
0x080	PC0X	256	Pre-computed point 0, x coordinate
0x0A0	PC0Y	256	Pre-computed point 0, y coordinate
0x0C0	PC1X	256	Pre-computed point 1, x coordinate
0x0E0	PC1Y	256	Pre-computed point 1, y coordinate
0x100	PC2X	256	Pre-computed point 2, x coordinate
0x120	PC2Y	256	Pre-computed point 2, y coordinate

0x140	PC3X	256	Pre-computed point 3, x coordinate
0x160	PC3Y	256	Pre-computed point 3, y coordinate
0x180	PC4X	256	Pre-computed point 4, x coordinate
0x1A0	PC4Y	256	Pre-computed point 4, y coordinate
0x1C0	PC5X	256	Pre-computed point 5, x coordinate
0x1E0	PC5Y	256	Pre-computed point 5, y coordinate
0x200	PC6X	256	Pre-computed point 6, x coordinate
0x220	PC6Y	256	Pre-computed point 6, y coordinate
0x240	PC7X	256	Pre-computed point 7, x coordinate
0x260	PC7Y	256	Pre-computed point 7, y coordinate
0x280	RA	256	Reserved
0x2A0	RB	256	Reserved
0x2C0	RC	256	Reserved
0x2E0	RD	256	Reserved

Total of **768** bytes must be reserved to store the Operation buffer.

10.6.2.2 Up to 384-bit operations (ECCSIZE = 2)

Table 10-7 - Operation Buffer Mapping (up to 384-bit operands)

Offset	Parameter	Size (bits)	Description
0x000	Mod P	384	P modulo
0x030	Mod N	384	N modulo
0x060	CA	384	ECC – CA parameter
0x090	CB	384	ECC – CB parameter
0x0C0	PC0X	384	Pre-computed point 0, x coordinate
0x0F0	PC0Y	384	Pre-computed point 0, y coordinate
0x120	PC1X	384	Pre-computed point 1, x coordinate
0x150	PC1Y	384	Pre-computed point 1, y coordinate
0x180	PC2X	384	Pre-computed point 2, x coordinate
0x1B0	PC2Y	384	Pre-computed point 2, y coordinate
0x1E0	PC3X	384	Pre-computed point 3, x coordinate
0x210	PC3Y	384	Pre-computed point 3, y coordinate
0x240	PC4X	384	Pre-computed point 4, x coordinate
0x270	PC4Y	384	Pre-computed point 4, y coordinate
0x2A0	PC5X	384	Pre-computed point 5, x coordinate
0x2D0	PC5Y	384	Pre-computed point 5, y coordinate
0x300	PC6X	384	Pre-computed point 6, x coordinate
0x330	PC6Y	384	Pre-computed point 6, y coordinate
0x360	PC7X	384	Pre-computed point 7, x coordinate
0x390	PC7Y	384	Pre-computed point 7, y coordinate
0x3C0	RA	384	Reserved
0x3F0	RB	384	Reserved
0x420	RC	384	Reserved
0x450	RD	384	Reserved

Total of **1152** bytes must be reserved to store the Operation buffer.

10.6.2.3 Up to 521-bit operations (ECCSIZE = 3)

Table 10-8 - Operation Buffer Mapping (up to 521-bit operands)

Offset	Parameter	Size (bits)	Description
0x000	Mod P	544	P modulo
0x044	Mod N	544	N modulo
0x088	CA	544	ECC – CA parameter

0x0CC	CB	544	ECC – CB parameter
0x110	PC0X	544	Pre-computed point 0, x coordinate
0x154	PC0Y	544	Pre-computed point 0, y coordinate
0x198	PC1X	544	Pre-computed point 1, x coordinate
0x1DC	PC1Y	544	Pre-computed point 1, y coordinate
0x220	PC2X	544	Pre-computed point 2, x coordinate
0x264	PC2Y	544	Pre-computed point 2, y coordinate
0x2A8	PC3X	544	Pre-computed point 3, x coordinate
0x2EC	PC3Y	544	Pre-computed point 3, y coordinate
0x330	PC4X	544	Pre-computed point 4, x coordinate
0x374	PC4Y	544	Pre-computed point 4, y coordinate
0x3B8	PC5X	544	Pre-computed point 5, x coordinate
0x3FC	PC5Y	544	Pre-computed point 5, y coordinate
0x440	PC6X	544	Pre-computed point 6, x coordinate
0x484	PC6Y	544	Pre-computed point 6, y coordinate
0x4C8	PC7X	544	Pre-computed point 7, x coordinate
0x50C	PC7Y	544	Pre-computed point 7, y coordinate
0x550	RA	544	Reserved
0x594	RB	544	Reserved
0x5D8	RC	544	Reserved
0x61C	RD	544	Reserved

Total of **1632** bytes must be reserved to store the Operation buffer.

10.6.3 Memory Allocation for SCA Data in System Memory

The system memory is used to store operands, results, and various SCA variables. The software (user) must reserve the needed space in the system memory. Pointers (memory addresses) to this space must be defined before to start any SCA operations. The data pointers must be stored in the following APB registers:

SCA_PPX_ADDR, SCA_PPY_ADDR, SCA_PPZ_ADDR, SCA_PQX_ADDR,
SCA_PQY_ADDR, SCA_PQZ_ADDR, SCA_RDSA_ADDR,
SCA_RES_ADDR, SCA_OP_BUFF_ADDR

Some SCA operations need extra system memory to store/load variables. This extra memory is defined as the operation buffer (OPBUFFER). The size of the **OPBUFFER** depends on the operand size (SCA_CN.ECCSIZE) and the used mode (SCA_CN.MANPARAM).

If **MANPARAM** is not used (SCA_CN.MANPARAM=0), SCA uses hardcoded parameters for **NIST P256** and a restricted OPBUFFER is needed. Otherwise (SCA_CN.MANPARAM=1), a larger OPBUFFER is needed to store curve parameters.

1) SCA_CN.MANPARAM=0 (**NIST P256**)

OPBUFFER is 4 blocks * ECCSIZE:

- ECCSIZE=256 → OPBUFFER = 4 * 256 = 6144 bits = 128 bytes
- ECCSIZE=384 → OPBUFFER = 4 * 384 = 9216 bits = 192 bytes
- ECCSIZE=521 → OPBUFFER = 4 * 544 = 13056 bits = 272 bytes

2) SCACN.MANPARAM=1

OPBUFFER is 24 blocks * ECCSIZE:

- ECCSIZE=256 → OPBUFFER = 24 * 256 = 6144 bits = 768 bytes
- ECCSIZE=384 → OPBUFFER = 24 * 384 = 9216 bits = 1152 bytes
- ECCSIZE=521 → OPBUFFER = 24 * 544 = 13056 bits = 1632 bytes

The following tables define which memory location shall be reserved for each SCA operation.

NA means that the corresponding memory is not used during the operation. Therefore, the specific address register (refer to SCA registers) is not used by the SCA.

10.6.3.1 OPCODE = 0: MM - Modular Multiplication

	Space to be allocated in system RAM (Bytes)									
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER	Total
256	32	32	NA	NA	NA	NA	32	NA	NA	96
384	48	48	NA	NA	NA	NA	48	NA	NA	144
521	65	65	NA	NA	NA	NA	65	NA	NA	195

10.6.3.2 OPCODE = 1: MD - Modular Division

	Space to be allocated in system RAM (Bytes)									
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER	Total
256	32	32	NA	NA	NA	NA	32	NA	128	224
384	48	48	NA	NA	NA	NA	48	NA	192	336
521	65	65	NA	NA	NA	NA	65	NA	272	467

10.6.3.3 OPCODE = 2: MA - Modular addition

	Space to be allocated in system RAM (Bytes)									
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER	Total
256	32	32	NA	NA	NA	NA	32	NA	NA	96
384	48	48	NA	NA	NA	NA	48	NA	NA	144
521	65	65	NA	NA	NA	NA	65	NA	NA	195

10.6.3.4 OPCODE = 3: MS - Modular subtraction

	Space to be allocated in system RAM (Bytes)									
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER	Total
256	32	32	NA	NA	NA	NA	32	NA	NA	96
384	48	48	NA	NA	NA	NA	48	NA	NA	144
521	65	65	NA	NA	NA	NA	65	NA	NA	195

10.6.3.5 OPCODE = 5: ECJAADD - ECC point operation, Jacobian + Affine

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	32	32	NA	96	NA	128	768	384	1024
384	48	48	48	48	48	NA	144	NA	192	1152	576	1536
521	65	65	65	65	65	NA	195	NA	272	1632	792	2152

10.6.3.6 OPCODE = 6: ECJASUB - ECC point operation, Jacobian – Affine

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	32	32	NA	96	NA	128	768	384	1024
384	48	48	48	48	48	NA	144	NA	192	1152	576	1536
521	65	65	65	65	65	NA	195	NA	272	1632	792	2152

10.6.3.7 OPCODE = 7: ECJDBL - ECC point operation, Jacobian double

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	NA	NA	NA	96	NA	128	768	320	960
384	48	48	48	NA	NA	NA	144	NA	192	1152	480	1440
521	65	65	65	NA	NA	NA	195	NA	272	1632	662	2022

10.6.3.8 OPCODE = 10: ECJACTOAF - ECC point operation, coordinates conversion

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	NA	NA	NA	96	NA	128	768	320	960
384	48	48	48	NA	NA	NA	144	NA	192	1152	480	1440
521	65	65	65	NA	NA	NA	195	NA	272	1632	662	2022

10.6.3.9 OPCODE = 11 : ECAFFTOJAC - ECC point operation, coordinates conversion

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	NA	NA	NA	NA	96	NA	128	768	288	928
384	48	48	NA	NA	NA	NA	144	NA	192	1152	432	1392
521	65	65	NA	NA	NA	NA	195	NA	272	1632	597	1957

10.6.3.10 OPCODE = 12: ECPVERIF - EC point verification

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	NA	NA	NA	96	NA	128	768	320	960
384	48	48	48	NA	NA	NA	144	NA	192	1152	480	1440
521	65	65	65	NA	NA	NA	195	NA	272	1632	662	2022

10.6.3.11 OPCODE = 13: ECSPSCAL - EC secure scalar for P256 (kP)

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	NA	NA	32	NA	NA	NA	96	NA	128	768	256	896
384	NA	NA	48	NA	NA	NA	144	NA	192	1152	384	1344
521	NA	NA	65	NA	NA	NA	195	NA	272	1632	532	1892

10.6.3.12 OPCODE = 14: ECSCALQ - EC secure scalar for Diffie Hellman operation (kQ)

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	32	32	32	96	NA	128	768	416	1056
384	48	48	48	48	48	48	144	NA	192	1152	624	1584
521	65	65	65	65	65	65	195	NA	272	1632	857	2217

10.6.3.13 OPCODE = 15: ECDSASIG - ECDSA signature

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	32	32	32	96	NA	128	768	416	1056
384	48	48	48	48	48	48	144	NA	192	1152	624	1584
521	65	65	65	65	65	65	195	NA	272	1632	857	2217

10.6.3.14 OPCODE = 16: ECDSAVER - ECDSA verification

Space to be allocated in system RAM (Bytes)												
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	32	32	32	96	32	128	768	448	1088
384	48	48	48	48	48	48	144	48	192	1152	672	1632
521	65	65	65	65	65	65	195	65	272	1632	922	2282

10.6.3.15 OPCODE = 18: ECSPC2kPA2vQ - EC unsecure 2kP + 2vQ for P256

	Space to be allocated in system RAM (Bytes)											
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	32	32	32	32	32	32	96	NA	128	768	416	1056
384	48	48	48	48	48	48	144	NA	192	1152	624	1584
521	65	65	65	65	65	65	195	NA	272	1632	857	2217

10.6.3.16 OPCODE = 19: EC P-Q - with hard coded P and result in affine

	Space to be allocated in system RAM (Bytes)											
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	NA	NA	NA	32	32	NA	96	NA	128	768	288	928
384	NA	NA	NA	48	48	NA	144	NA	192	1152	432	1392
521	NA	NA	NA	65	65	NA	195	NA	272	1632	597	1957

10.6.3.17 OPCODE = 20: EC P+Q - with hard coded P and result in affine

	Space to be allocated in system RAM (Bytes)											
ECCSIZE	PPX	PPY	PPZ	PQX	PQY	PQZ	RES	RDSA	OPBUFFER		Total	
									MANPARAM		MANPARAM	
									0	1	0	1
256	NA	NA	NA	32	32	NA	96	NA	128	768	288	928
384	NA	NA	NA	48	48	NA	144	NA	192	1152	432	1392
521	NA	NA	NA	65	65	NA	195	NA	272	1632	597	1957

10.7 Cryptographic Registers

Address assignments for the Cryptographic Accelerator registers are outlined in Table 10-9.

Table 10-9. Cryptographic Accelerator Register Address (Base ADDR = 0x4000_1000)

Offset	Access	Register	Description
0x0000	RW	CRYPTO_CTRL	Crypto Control Register
0x0004	RW	CIPHER_CTRL	Cipher Control Register
0x0008	RW	HASH_CTRL	HASH Control Register
0x000C	RW	CRC_CTRL	CRC Control Register
0x0010	RW	DMA_SRC	Crypto DMA Source Register
0x0014	RW	DMA_DEST	Crypto DMA Destination Register
0x0018	RW	DMA_CNT	Crypto DMA Count Register
0x001C	R	RFU	Reserved
0x0020 ... 0x002C	W	CRYPTO_DIN_[3:0]	Crypto Data In Register [3:0]
0x0030 ... 0x003C	R	CRYPTO_DOUT_[3:0]	Crypto Data Out Register [3:0]
0x0040	RW	CRC_POLY	CRC Polynomial Register
0x0044	RW	CRC_VAL	CRC Value Register
0x0048	R	CRC_PRNG	Pseudo-Random Number Generator Register
0x004C	RW	HAM_ECC	Hamming ECC Register
0x0050 ... 0x005C	RW	CIPHER_INIT_[3:0]	Cipher Initial Vector Register [3:0]
0x0060 ... 0x007C	W	CIPHER_KEY_[7:0]	Cipher Key Register [7:0]
0x0080 ... 0x00BC	RW	HASH_DIGEST_[15:0]	HASH Message Digest Register [15:0]
0x00C0... 0x00CC	RW	HASH_MSG_SZ_[3:0]	HASH Message Size Register [3:0]
0x00D0 – 0x00D4	RW	A_LENGTH_[1:0]	AAD Length Register [1:0]
0x00D8 – 0x00DC	RW	PLD_LENGTH_[1:0]	PLD Length Register [1:0]
0x00E0 ... 0x00EC	RW	TAGMIC_[0:3]	Tag/Mic Register [3:0]
0x00F0 ... 0x00FC	R	RFU	Reserved
0x0100	RW	SCA_CN	SCA Control Register
0x0104	RW	SCA_ACN	SCA Advanced Control Register
0x0108	RW	SCA_ST	SCA Status Register
0x010C	RW	SCA_PPX_ADDR	PPX Address Register
0x0110	RW	SCA_PPY_ADDR	PPY Address Register
0x0114	RW	SCA_PPZ_ADDR	PPZ Address Register
0x0118	RW	SCA_PQX_ADDR	PQX Address Register
0x011C	RW	SCA_PQY_ADDR	PQY Address Register
0x0120	RW	SCA_PQZ_ADDR	PQZ Address Register
0x0124	RW	SCA_RDSA_ADDR	R Portion for ECDSA Signature Verification Register
0x0128	RW	SCA_RES_ADDR	Result Address Register
0x012C	RW	SCA_OP_BUFF_ADDR	Operation Buffer Address
0x0130	RW	SCA_MODDATA	Modulo Data Input Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

10.7.1 Crypto Control Register (CRYPTO_CTRL, Offset 0x0000)

Table 10-10. CRYPTO_CTRL (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	DONE	RDY	ERR	RFU	CPH_DONE	HSH_DONE	GLS_DONE	DMA_DONE
Reset	0	1	0	0	0	0	0	0
Access	R	R	R	R*	RW/RW1C*	RW/RW1C*	RW/RW1C*	RW/RW1C*
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	DMADNEMSK	FLAG_MODE	RFU[1:0]		RDSRC[1:0]		WRSRC[1:0]	
Reset	0	0	00		00		00	
Access	RW	RW	R		RW		RW	
Position	7	6	5	4	3	2	1	0
Field	WAIT_POL	WAIT_EN	BSI	BSO	RFU	SRC	INT	RST
Reset	0	0	0	0	0	0	0	0
Access	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bits	Description	Settings
RST	0	Reset. This bit is used to reset the crypto accelerator. All crypto internal states and related registers are reset to their default reset values. Control register such as CRYPTO_CTRL, CIPHER_CTRL, HASH_CTRL, CRC_CTRL. HASH_MSG_SZ[3:0] retains its values. This bit automatically clears itself after one cycle.	0: No effect 1: Reset crypto accelerator
INT	1	Interrupt Enable. Generates an interrupt when done or error set.	0: Interrupt disabled. 1: Interrupt enabled.
SRC	2	Source Select. This bit selects the hash function and CRC generator input source.	0: Input FIFO 1: Output FIFO
RFU	3	Reserved	N/A
BSO	4	Byte Swap Output. Note. No byte swap occurs if there is not a full word.	0: No effect. 1: Byte swap output.
BSI	5	Byte Swap Input. Note. No byte swap occurs if there is not a full word.	0: No effect. 1: Byte swap input.
WAIT_EN	6	Wait Pin Enable. This can be used to hold off the crypto DMA until an external memory is ready. This is useful for transferring pages from NAND flash which may take several microseconds to become ready.	0: Disabled 1: Enabled
WAIT_POL	7	Wait Pin Polarity. When the wait pin is enabled, this bit selects its active state.	0: Active low 1: Active high
WRSRC	9:8	Write FIFO Source Select. This field determines where data written to the write FIFO comes from. When data is written to the write FIFO, it is always written out the DMA. To decrypt or encrypt data, the write FIFO source should be set to the cipher output. To implement memcpy() or memset() functions, or to fill memory with random data, the write FIFO source should be set to the read FIFO. When calculating a HASH or CMAC, the write FIFO should be disabled.	00: None 01: Cipher Output 10: Read FIFO 11: Reserved

Name	Bits	Description	Settings
RDSRC	11:10	Read FIFO Source Select. This field selects the source of the read FIFO. Typically, it is set to use the DMA. To implement a memset() function, the read FIFO DMA should be disabled. To fill memory with random data or to hash random numbers, the read FIFO source should be set to the random number generator.	00: DMA disabled 01: DMA or APB 10: RNG 11: Reserved
RFU	13:12	Reserved	N/A
FLAG_MODE	14	Done Flag Mode. This bit configures the access behavior of the individual CRYPTO_CTRL Done flags (CRYPTO_CTRL[27:24]). This bit is cleared only on reset to limit upkeep, i.e. once set, it remains so until a reset occurs.	0: Unrestricted write(0 or 1) of CRYPTO_CTRL[27:24] flags 1: Access to CRYPTO_CTRL[27:24] are "write 1 to clear/write 0 no effect"
DMADNEMSK	15	DMA Done Flag Mask. This bit masks the DMA_DONE flag from being used to generate the CRYPTO_CTRL.DONE flag, and thus disables a DMA_DONE condition from generating an interrupt. The DMA_DONE flag itself is unaffected and still may be monitored. This allows more optimal interrupt-driven crypto operations using DMA.	0: DMA_DONE not used in setting CRYPTO_CTRL.DONE bit. 1: DMA_DONE used in setting CRYPTO_CTRL.DONE bit.
RFU	23:16	Reserved	N/A
DMA_DONE	24	DMA Done. DMA write/read operation is complete. This bit must be cleared before starting a DMA operation.	0: Not Done 1: Operation Done (See Note)
GLS_DONE	25	Galois Done. FIFO is full and CRC or Hamming Code Generator is enabled. This bit must be cleared before starting a CRC operation	0: Not Done 1: Operation Done (See Note)
HSH_DONE	26	Hash Done. SHA operation is complete. This bit must be cleared before starting a HASH operation.	0: Not Done 1: Operation Done (See Note)
CPH_DONE	27	Cipher Done. Either AES or DES encryption/decryption operation is complete. This bit must be cleared before starting a cipher operation.	0: Not Done 1: Operation Done (See Note)
RFU	28	Reserved	N/A
ERR	29	AHB Bus Error. This bit is set when the DMA encounters a bus error during a read or write operation. Once this bit is set, the DMA stops. This bit can only be cleared by resetting the crypto block.	0: No Error 1: Error
RDY	30	Ready. Crypto block ready for more data	0: crypto busy 1: ready for data
DONE	31	Done. One or more cryptographic calculations complete. (logical 'OR' of done flags). See the CRYPTO_CTRL.DMADNEMSK bit description further detail.	0: Not Done 1: Operation Done

Note: If FLAG_MODE = 0, the DMA_DONE, GLS_DONE, HSH_DONE and CPH_DONE bits functionality allows each flag to be written to either 0 or 1. If FLAG_MODE = 1, the DMA_DONE, GLS_DONE, HSH_DONE and CPH_DONE bits are cleared when writing a 1 to that flag, and writing a 0 has no effect on that flag.

In addition, for cipher, hash, or Galois operation, when RDSRC is configured for DMA, the associated DONE flags only set after the entire DMA operation is complete with the end cipher, hash or Galois block of data, i.e. intermediate operation on cipher, hash or Galois blocks do not continually set the associated DONE flag. For cipher, when WRSRC is configured for cipher output, the CPH_DONE sets only after the last cipher text has completed the DMA transfer out to memory. Therefore, when using the crypto DMA for these operations, the DMA_DONE does not normally need processing and can be left masked from interrupting the CPU.

10.7.2 Cipher Control Register (CIPHER_CTRL, Offset 0x0004)

Table 10-11. CIPHER_CTRL (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[4:0]				CCML			
Reset	0000 0				000			
Access	R				RW			
Position	15	14	13	12	11	10	9	8
Field	CCMM[2:0]			DTYPE	COMPH	MODE[2:0]		
Reset	000			0	0	0		
Access	R			RW	RW	RW		
Position	7	6	5	4	3	2	1	0
Field	RFU	CIPHER[2:0]			SRC[1:0]		KEY	ENC
Reset	0	000			00		0	0
Access	R	RW			RW		RW	RW

Name	Bits	Description	Settings
ENC	0	Encrypt. Select encryption or decryption of input data.	0: Encrypt 1: Decrypt
KEY	1	Load Key from crypto DMA. This bit is automatically cleared by hardware after the DMA has completed loading the key. When the DMA operation is done, it sets the appropriate crypto DMA Done flag.	0: NOP 1: Initiate key loading from DMA
SRC	3:2	Source of Random key.	0x: User cipher key (0x4000_1060) 10: AES_KEY0 and AES_KEY1 from AES (0x4000_5000 to 0x4000_501F) 11: PUF_KEY1
CIPHER	6:4	Cipher Operation Select. Symmetric Block Cipher algorithm selection or memory operation.	000: Disabled 001: AES-128 010: AES-192 011: AES-256 100: DES 101: TDEA
RFU	7	Reserved	N/A
MODE	10:8	Mode Select. Mode of operation for block cipher or memory operation.	000: ECB 001: CBC 010: CFB 011: OFB 100: CTR 101: GCM 110: CCM
COMPH	11	H Vector Computation. This bit is automatically cleared by hardware.	0: NOP 1: Starts an H vector computation
DTYPE	12	GCM/CCM data type	0: AAD 1: Payload

Name	Bits	Description	Settings
CCMM	15:13	CCM M parameter	000: Reserved 001: 4 010: 6 011: 8 100: 10 101: 12 110: 14 111: 16
CCML	18:16	CCM L parameter	000: Reserved 001: 2 010: 3 011: 4 100: 5 101: 6 110: 7 111: 8
RFU	31:19	Reserved	N/A

10.7.3 HASH Control Register (HASH_CTRL, Offset 0x0008)

Table 10-12. HASH_CTRL (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU	RFU	LAST	HASH[2:0]			XOR	INIT
Reset	0	0	0	000			0	0
Access	R	R	RW	RW			RW	RW

Name	Bits	Description	Settings
INIT	0	Initialize. Initializes hash registers with standard constants.	0: NOP 1: Initialize Hash Values.
XOR	1	XOR. XOR data with IV from cipher block. Useful when calculating HMAC to XOR the input pad and output pad.	0 - no XOR 1 - XOR input with IV
HASH	4:2	Hash function selection.	000 - HASH disabled 001 - SHA-1 010 - SHA-224 011 - SHA-256 100 - SHA-384 101 - SHA-512 Others: Reserved.
LAST	5	Last Message Bit. This bit shall be set along with the HASH_MSG_SZ register prior to hashing the last 512 or 1024-bit block of the message data. It allows automatic preprocessing of the last message padding, which includes the trailing bit "1", followed by the respective number of zero bits for the last block size and finally the message length represented in bytes. The bit is automatically cleared at the same time the HASH DONE is set, designating the completion of the last message hash.	0: no effect 1: Last message data.
RFU	31:6	Reserved	N/A

Note: The automatic padding feature can only be used in terms of message bytes, not bits. Therefore, the HASH_MSG_SZ registers are in terms of bytes. In addition, the feature automatically generates an additional "padding-only" block, if the last block of message data cannot accommodate the 64 or 128-bit padding block.

As an exception, attempting to hash a "0" message size block must include a "dummy" write to the HASH message digest register.

10.7.4 CRC Control Register (CRC_CTRL, Offset 0x000C)

Table 10-13. CRC_CTRL (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU	RFU	HRST	HAM	ENT	PRNG	MSB	CRC
Reset	0	0	0	0	0	0	0	0
Access	R	R	W	RW	RW	RW	RW	RW

Name	Bits	Description	Settings
CRC	0	Cyclic Redundancy Check Enable. The CRC cannot be enabled if the PRNG is enabled.	0: CRC disabled 1: CRC enabled
MSB	1	MSB select. This bit selects the order of calculating CRC on data.	0: LSB data first 1: MSB data first
PRNG	2	Pseudo Random Number Generator Enable. If entropy is disabled, this outputs one byte of pseudo random data per clock cycle. If entropy is enabled, data is output at a rate of one bit per clock cycle.	0: PRNG disabled 1: PRNG enabled
ENT	3	Entropy Enable. If the PRNG is enabled, this mixes the high frequency ring oscillator with the LFSR. If the PRNG is disabled, the raw entropy data is output at a rate of 1 bit per clock. This makes it possible to characterize the quality of the entropy source.	0: No entropy 1: Add entropy
HAM	4	Hamming Code Enable. Enable hamming code calculation	0: Hamming disabled 1: Hamming enabled
HRST	5	Hamming Reset. Reset Hamming code ECC generator for next block	0: NOP 1: Reset Hamming Register
RFU	31:6	Reserved	N/A

10.7.5 Crypto DMA Source Register (DMA_SRC, Offset 0x0010)

Table 10-14. DMA_SRC (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	ADDR[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	ADDR[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	ADDR[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
ADDR	31:0	DMA Source Address.	

10.7.6 Crypto DMA Destination Register (DMA_DEST, Offset 0x0014)

Table 10-15. DMA_DEST (Offset 0x0014)

Position	31	30	29	28	27	26	25	24
Field	ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	ADDR[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	ADDR[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	ADDR[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
ADDR	31:0	DMA Destination Address.	

10.7.7 Crypto DMA Count Register (DMA_CNT, Offset 0x0018)

Table 10-16. DMA_CNT (Offset 0x0018)

Position	31	30	29	28	27	26	25	24
Field	CNT[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	CNT[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	CNT[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	CNT[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
CNT	31:0	DMA Byte Count.	

10.7.8 Crypto Data In Register 0 (CRYPTO_DIN_0, Offset 0x0020)

Crypto Data In Registers are write only and reading of any of these registers return 0.

Table 10-17. CRYPTO_DIN_0 (Offset 0x0020)

Position	31	30	29	28	27	26	25	24
Field	DATA[31:24]							
Reset	0000 0000							
Access	W							
Position	23	22	21	20	19	18	17	16
Field	DATA[23:16]							
Reset	0000 0000							
Access	W							
Position	15	14	13	12	11	10	9	8
Field	DATA[15:8]							
Reset	0000 0000							
Access	W							
Position	7	6	5	4	3	2	1	0
Field	DATA[7:0]							
Reset	0000 0000							
Access	W							

Name	Bits	Description	Settings
DATA	31:0	Crypto Data Input. Data input can be written to this register instead of using the DMA. This register writes to the FIFO. This register occupies four successive words to allow the use of multi-store instructions. Words can be written to any location, they are placed in the FIFO in the order they are written. The endian swap input control bit (CRYPTO_CTRL.BSI) affects this register.	

10.7.9 Crypto Data In Register [3:1]

For details, refer to Crypto Data In Register 0.

Register	Offset	Description
CRYPTO_DIN_1	0x0024	DATA[63:32]
CRYPTO_DIN_2	0x0028	DATA[95:64]
CRYPTO_DIN_3	0x002C	DATA[127:96]

10.7.10 Crypto Data Out Register 0 (CRYPTO_DOUT_0, Offset 0x0030)

Table 10-18. CRYPTO DOUT 0 (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	DATA[31:24]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	DATA[23:16]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	DATA[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	DATA[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
DATA	31:0	Crypto Data Output. Resulting data from cipher calculation. Data is placed in the lower words of these four registers depending on the algorithm. For block cipher modes, this register holds the result of most recent encryption or decryption operation. These registers are affected by the endian swap bit (CRYPTO_CTRL.BSO).	

10.7.11 Crypto Data Out Register [3:1]

For details, refer to Crypto Data Out Register 0.

Register	Offset	Description
CRYPTO_DOUT_1	0x0034	DATA[63:32]
CRYPTO_DOUT_2	0x0038	DATA[95:64]
CRYPTO_DOUT_3	0x003C	DATA[127:96]

10.7.12 CRC Polynomial Register (CRC_POLY, Offset 0x0040)

Table 10-19. CRC POLY (Offset 0x0040)

Position	31	30	29	28	27	26	25	24
Field	POLY[31:24]							
Reset	1110 1101							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	POLY[23:16]							
Reset	1011 1000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	POLY[15:8]							
Reset	1000 0011							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	POLY[7:0]							
Reset	0010 0000							
Access	RW							

Name	Bits	Description	Settings
POLY	31:0	CRC Polynomial. The polynomial to be used for Galois Field calculations (CRC or LFSR) should be written to this register. This register is affected by the MSB control bit.	

10.7.13 CRC Value Register (CRC_VAL, Offset 0x0044)

Table 10-20. CRC_VAL (Offset 0x0044)

Position	31	30	29	28	27	26	25	24
Field	VAL[31:24]							
Reset	1111 1111							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	VAL[23:16]							
Reset	1111 1111							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	VAL[15:8]							
Reset	1111 1111							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	VAL[7:0]							
Reset	1111 1111							
Access	RW							

Name	Bits	Description	Settings
VAL	31:0	CRC Value. This is the state for the Galois Field. This register holds the result of a CRC calculation or the current state of the LFSR. This register is affected by the MSB control bit.	

10.7.14 Pseudo-Random Number Generator Register (CRC_PRNG, Offset 0x0048)

Table 10-21. CRC PRNG (Offset 0x0048)

Position	31	30	29	28	27	26	25	24
Field	PRNG[31:24]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	PRNG[23:16]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	PRNG[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	PRNG[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
PRNG	31:0	Pseudo Random Value. Output of the Galois Field shift register. This holds the resulting pseudo-random number if entropy is disabled or true random number if entropy is enabled.	

10.7.15 Hamming ECC Register (HAM_ECC, Offset 0x004C)

Table 10-22. HAM ECC (Offset 0x004C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[6:0]							PAR
Reset	000 0000							0
Access	R							RW
Position	15	14	13	12	11	10	9	8
Field	ECC[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	ECC[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
ECC	15:0	Hamming ECC Value. These bits are the even parity of their corresponding bit groups.	
PAR	16	Parity. This is the parity of the entire array.	0: Even parity 1: Odd parity
RFU	31:17	Reserved	N/A

10.7.16 Cipher Initial Vector Register 0 (CIPHER_INIT_0, Offset 0x0050)

Table 10-23. CIPHER_INIT_0 (Offset 0x0050)

Position	31	30	29	28	27	26	25	24
Field	IVEC[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	IVEC[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	IVEC[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	IVEC[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
IVEC	31:0	Initial Vector. For block cipher operations that use CBC, CFB, OFB, or CNTR modes, this register holds the initial value. This register is updated with each encryption or decryption operation. This register is affected by the endian swap bits.	

10.7.17 Cipher Initial Vector Register [3:1]

For details, refer to Cipher Initial Vector Register 0.

Register	Offset	Description
CIPHER_INIT_1	0x0054	IVEC[63:32]
CIPHER_INIT_2	0x0058	IVEC[95:64]
CIPHER_INIT_3	0x005C	IVEC[127:96]

10.7.18 Cipher Key Register 0 (CIPHER_KEY_0, Offset 0x0060)

Cipher Key Registers are write only and reading of any of these registers return 0.

Table 10-24. CIPHER_KEY_0 (Offset 0x0060)

Position	31	30	29	28	27	26	25	24
Field	KEY[31:24]							
Reset	0000 0000							
Access	W							
Position	23	22	21	20	19	18	17	16
Field	KEY[23:16]							
Reset	0000 0000							
Access	W							
Position	15	14	13	12	11	10	9	8
Field	KEY[15:8]							
Reset	0000 0000							
Access	W							
Position	7	6	5	4	3	2	1	0
Field	KEY[7:0]							
Reset	0000 0000							
Access	W							

Name	Bits	Description	Settings
KEY	31:0	Cipher Key. This register holds the key used for block cipher operations. The lower words are used for block ciphers that use shorter key lengths. This register is affected by the endian swap input control bits.	

10.7.19 Cipher Key Register [7:1]

For details, refer to Cipher Key Register 0.

Register	Offset	Description
CIPHER_KEY_1	0x0064	KEY[63:32]
CIPHER_KEY_2	0x0068	KEY[95:64]
CIPHER_KEY_3	0x006C	KEY[127:96]
CIPHER_KEY_4	0x0070	KEY[159:128]
CIPHER_KEY_5	0x0074	KEY[191:160]
CIPHER_KEY_6	0x0078	KEY[223:192]
CIPHER_KEY_7	0x007C	KEY[255:224]

10.7.20 HASH Message Digest Register 0 (HASH_DIGEST_0, Offset 0x0080)

Table 10-25. HASH_DIGEST_0 (Offset 0x0080)

Position	31	30	29	28	27	26	25	24
Field	HASH[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	HASH[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	HASH[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	HASH[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
HASH	31:0	This register holds the calculated hash value. This register is affected by the endian swap bits.	

10.7.21 HASH Digest Register [15:1]

For details, refer to HASH Digest Register 0.

Register	Offset	Description
HASH_DIGEST_1	0x0084	HASH[63:32]
HASH_DIGEST_2	0x0088	HASH[95:64]
HASH_DIGEST_3	0x008C	HASH[127:96]
HASH_DIGEST_4	0x0090	HASH[159:128]
HASH_DIGEST_5	0x0094	HASH[191:160]
HASH_DIGEST_6	0x0098	HASH[223:192]
HASH_DIGEST_7	0x009C	HASH[255:224]
HASH_DIGEST_8	0x00A0	HASH[287:256]
HASH_DIGEST_9	0x00A4	HASH[319:288]
HASH_DIGEST_10	0x00A8	HASH[351:320]
HASH_DIGEST_11	0x00AC	HASH[383:352]
HASH_DIGEST_12	0x00B0	HASH[415:384]
HASH_DIGEST_13	0x00B4	HASH[447:416]
HASH_DIGEST_14	0x00B8	HASH[479:448]
HASH_DIGEST_15	0x00BC	HASH[511:480]

10.7.22 HASH Message Size Register 0 (HASH_MSG_SZ_0, Offset 0x00C0)

Table 10-26. HASH_MSG_SZ_0 (Offset 0x00C0)

Position	31	30	29	28	27	26	25	24
Field	MSGSZ[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	MSGSZ[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	MSGSZ[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	MSGSZ[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
MSGSZ	31:0	Message Size. This register holds the lowest 32-bit of message size in bytes.	

10.7.23 HASH Message Size Register [3:1]

For details, refer to HASH Message Size Register 0.

Register	Offset	Description
HASH_MSG_SZ_1	0x00C4	MSGSZ[63:32]
HASH_MSG_SZ_2	0x00C8	MSGSZ [95:64]
HASH_MSG_SZ_3	0x00CC	MSGSZ [124:96]

10.7.24 AAD Length Register 0 (A_LENGTH_0, Offset 0x00D0)

Table 10-27. A_LENGTH_0 (Offset 0x00D0)

Position	31	30	29	28	27	26	25	24
Field	A_LENGTH[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	A_LENGTH[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	A_LENGTH[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	A_LENGTH[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
A_LENGTH[31:0]	31:0	AAD Length in bytes for AES GCM and CCM operations.	

10.7.25 AAD_Length Register 1

For details, refer to AAD Length Register 0.

Register	Offset	Description
A_LENGTH_1	0x00D4	A_LENGTH[63:32]

10.7.26 PLD Length Register 0 (PLD_LENGTH_0, Offset 0x00D8)

Table 10-28. PLD_LENGTH_0 (Offset 0x00D8)

Position	31	30	29	28	27	26	25	24
Field	PLD_LENGTH[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	PLD_LENGTH[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	PLD_LENGTH[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	PLD_LENGTH[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
PLD_LENGTH[31:0]	31:0	Payload Length in Bytes for AES, GCM, and CCM operations.	

10.7.27 PLD_Length Register 1

Refer to PLD Length Register 0 for details.

Register	Offset	Description
PLD_LENGTH_1	0x00DC	PLD_LENGTH[63:32]

10.7.28 TAG/MIC Register 0 (TAGMIC_0, Offset 0x00E0)

Table 10-29. TAGMIC 0 (Offset 0x00E0)

Position	31	30	29	28	27	26	25	24
Field	TAGMIC[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	TAGMIC[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	TAGMIC[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	TAGMIC[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
TAGMIC[31:0]	31:0	TAG/MIC output for AES GCM and CCM operations	

10.7.29 TAGMIC Register [3:1]

Refer to TAGMIC Register 0 for details.

Register	Offset	Description
TAGMIC_1	0x00E4	TAGMIC[63:32]
TAGMIC_2	0x00E8	TAGMIC[95:64]
TAGMIC_3	0x00EC	TAGMIC[127:96]

10.7.30 SCA Control Register (SCA_CN, Offset 0x0100)

Table 10-30. SCA_CN (Offset 0x0100)

Position	31	30	29	28	27	26	25	24
Field	RFU[5:0]						ECCSIZE[1:0]	
Reset	0000 00						00	
Access	R						RW	
Position	23	22	21	20	19	18	17	16
Field	RFU[2:0]			MODADDR[4:0]				
Reset	000			0 0000				
Access	R			RW				
Position	15	14	13	12	11	10	9	8
Field	RFU[2:0]			OPCODE[4:0]				
Reset	000			0 0000				
Access	R			RW				
Position	7	6	5	4	3	2	1	0
Field	RFU	HWKEY	MANPARAM	ERMEM	RFU	ABORT	SCAIE	STC
Reset	0	0	0	0	0	0	0	0
Access	R	RW	RW	RW	R	RW	RW	RW

Name	Bits	Description	Settings
STC	0	Start Calculation. STC function as both the control and the status of the SCA. Setting this bit to 1 activates the SCA to perform one of the functions defined by OPCODE bits. If an error is detected (PVF, FSMERR, COMPERR in SCAST register) the STC bit is immediately cleared by the hardware. Otherwise, the STC bit is automatically cleared following the completion of the calculation. Clearing the STC bit resets the controller to its default state.	0: Reset controller to its default state 1: Start Calculation
SCAIE	1	SCA Interrupt Enable.	0: Interrupt disabled 1: Interrupt enabled
ABORT	2	Abort ongoing operation. Automatically cleared by hardware upon abort completion.	0: No Effect 1: Abort
RFU	3	Reserved	N/A
ERMEM	4	Erase Crypto Memory. Set this bit to Erase the crypto memory defined by *_ADDR registers and ECCSIZE . Ongoing operation is automatically aborted before to start SCA memory erasing. Automatically cleared by hardware	0: No action 1: Erase SCA memory
MANPARAM	5	If cleared, ECC operations are performed by using NIST P256 hardcoded parameters. If set, ECC curve parameters must be filled in OPBUFF For more details refer to 10.6.2	0: Automatic NIST P256 parameters 1: Manual ECC curve parameters
HWKEY	6	Hardware Key. To select hardware key as private key for ECDSA signature and ECC scalar operations. 256-bits only available if SCA_ACN.MAN=0	0: Software key is used (PQx) 1: Hardware key is used. (PUF_KEY2)
RFU	7	Reserved	N/A
OPCODE	12:8	Op Code. These bits are used to define the operation to perform at STC=1	Refer to Table 10-4 and Table 10-5
RFU	15:13	Reserved	N/A

Name	Bits	Description	Settings
MODADDR	20:16	MODULO addressing. MODULO address must be set before writing a new 32-bit portion of the MODULO through ECDSADATA register	0: 32-bit LSB
RFU	23:21	Reserved	N/A
ECCSIZE	25:24	Define the maxim number of bits for SCA operation. SCA operation is slower for high ECCSIZE These bits are only usable if MANPARAM is set, otherwise the ECCSIZE is forced to '01' (256-bits) for P256 operations.	00: 256 bits 01: 256 bits 10: 384 bits 11: 521 bits
RFU	31:26	Reserved	N/A

10.7.31 SCA Advanced Control Register (SCA_ACN, Offset 0x0104)

Table 10-31. SCA_ACN (Offset 0x0104)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[5:0]						CARRYPOS[9:8]	
Reset	0000 00						00	
Access	R						RW	
Position	15	14	13	12	11	10	9	8
Field	CARRYPOS[7:0]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			RESSELECT		PLUSONE	AUTOCARRY	MAN
Reset	000			00		0	0	0
Access	R			RW		RW	RW	RW

Name	Bits	Description	Settings
MAN	0	SCA Manual Mode	0: Auto Mode 1: Manual Mode
AUTOCARRY	1	Automatically propagate the carry for the next operation. When this bit is set, SCACN.PLUSONE is atomically set on carry detection. If this bit is not set the SW is in charge to write SCACN.PLUSONE if needed. No effect if MAN is not set.	0: Software has to write to SCACN.PLUSONE 1: Autocarry
PLUSONE	2	Enable Carry propagation for the next operation. Automatically cleared to 0 if MAN is not set.	0: No Effect 1: Propagate Carry
RESSELECT	4:3	ALU Selection Automatically cleared to 0 in MAN is not set.	00: Auto 01: Result ALU_1 10: Result ALU_2 11: Auto
RFU	7:5	Reserved	N/A
CARRYPOS	17:8	To set Carry location. Can be used if operands are less or equal to 521 bits. According to the operand size CARRYPOS range is from 1 to 522. SCAST.CARRY is set when a carry is detected upon operation completion, based on the carry location defined by these bits. No effect if MAN is not set. Note: If operands are less than 256 bit, e.g. m bits with $m < 256$, the results are on $> m$ bits (e.g. $m+1$ for add, up to $2m$ for multiplication), with result ≤ 256 bits	
RFU	31:18	Reserved	N/A

10.7.32 SCA Status Register (SCA_ST, Offset 0x0008)

Table 10-32. SCA_ST (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				ALUNEG2	ALUNEG1	GTE2I2	CARRY
Reset	0000				0	0	0	0
Access	R				R	R	R	R
Position	7	6	5	4	3	2	1	0
Field	RFU	MEMERR	COMPERR	FSMERR	PVF2	PVF1	SCAIF	BUSY
Reset	0	0	0	0	0	0	0	0
Access	R	R	RW1C	RW1C	RW1C	RW1C	RW1C	RW

Name	Bits	Description	Settings
BUSY	0	SCA is busy. This bit is the image of STC bit in SCA_CN	0: SCA is available 1: SCA is busy
SCAIF	1	SCA interrupt Flag. The SCAIF bit is set by hardware when the SCA has completed an operation. This bit is cleared when writing a 1 to this bit, writing a 0 has no effect.	0: Interrupt has not occurred 1: Interrupt has occurred
PVF1	2	Point Verification Fail 1. This bit is set if the result of the EC point verification fails (refer to ECC operation table) IT is generated if ECC verification fails. This bit is cleared when writing a 1 to this bit, writing a 0 has no effect.	0: ECC Point Verification OK 1: ECC Point Verification has failed
PVF2	3	Point Verification Fail 2. This bit is set if the result of the EC point verification fails (refer to ECC operation table) IT is generated if ECC verification fails. This bit is cleared when writing a 1 to this bit, writing a 0 has no effect.	0: ECC Point Verification OK 1: ECC Point Verification has failed
FSMERR	4	FSM transition error can occur upon external fault injection. IT is generated. This bit is cleared when writing a 1 to this bit, writing a 0 has no effect.	0: FSM transition is OK 1: FSM transition error
COMPERR	5	EC Computation error. Ongoing operation has failed, severity depends on the processed operation. IT is generated. This bit is cleared when writing a 1 to this bit, writing a 0 has no effect.	0: No error 1: EC operation Error

Name	Bits	Description	Settings
MEMERR	6	Memory Access Error. This error flag is set when the SCA memory access is not authorized. e.g. SCAPPX_ADDR = 0x00000000, 0x00000000 is ROM memory, thus this location cannot be written, so SCA write to this location fails and MEMERR is set.	0: No error 1: Memory Error
RFU	7	Reserved	N/A
CARRY	8	Carry on ongoing operation	0: No carry 1: Carry
GTE2I2	9	This bit is set if the operation result is equal to above modulo x 2. Automatically cleared to 0 if SCA_ACN.MAN is not set.	0: Result < 2 x mod 1: Result ≥ 2 x mod
ALUNEG1	10	Sign of the subtraction result for ALU_1	0: ALU_1 result ≥ 0 1: ALU_1 result < 0
ALUNEG2	11	Sign of the subtraction result for ALU_2	0: ALU_2 result ≥ 0 1: ALU_2 result < 0
RFU	31:12	Reserved	N/A

10.7.33 Point P Address Register (SCA_PPi_ADDR, Offset 0x010C, 0x0110 and 0x0114)

Registers in which the address to PPX, PPY, and PPZ are stored.

Table 10-33. SCA_PPi_ADDR (Offset 0x010C, 0x0110 and 0x0114)

Position	31	30	29	28	27	26	25	24
Field	PPi_ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	PPi_ADDR [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	PPi_ADDR [15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	PPi_ADDR [7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
PPi_ADDR	31:0	Address where PPi (i = X, Y or Z) is stored in system memory.	

10.7.34 Point Q Address Register (SCA_PQi_ADDR, Offset 0x0118, 0x011C and 0x0120)

Registers in which the address to PQX, PQY, and PQZ are stored.

Table 10-34. SCA_PQi_ADDR (Offset 0x0118, 0x011C and 0x0120)

Position	31	30	29	28	27	26	25	24
Field	PQi_ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	PQi_ADDR [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	PQi_ADDR [15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	PQi_ADDR [7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
PQi_ADDR	31:0	Address where PQi (i = X, Y or Z) is stored in system memory.	

10.7.35 RDSA Address Register (SCA_RDSA_ADDR, Offset 0x0124)

Table 10-35. SCA_RDSA_ADDR (Offset 0x0124)

Position	31	30	29	28	27	26	25	24
Field	RDSA_ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	RDSA_ADDR [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	RDSA_ADDR [15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	RDSA_ADDR [7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
RDSA_ADDR	31:0	Address to store RDSA in system memory. RDSA is the R portion of the R, S ECDSA signature. This parameter is only used for ECDSA verification (OPCODE = 16)	

10.7.36 Result Address Register (SCA_RES_ADDR, Offset 0x0128)

Table 10-36. SCA_RES_ADDR (Offset 0x0128)

Position	31	30	29	28	27	26	25	24
Field	RES_ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	RES_ADDR [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	RES_ADDR [15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	RES_ADDR [7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
RES_ADDR	31:0	Address to store Result in system memory	

10.7.37 Operation Buffer Address Register (SCA_OP_BUFF_ADDR, Offset 0x012C)

Table 10-37. SCA_OP_BUFF_ADDR (Offset 0x012C)

Position	31	30	29	28	27	26	25	24
Field	OPBUFF_ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	OPBUFF_ADDR [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	OPBUFF_ADDR [15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	OPBUFF_ADDR [7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
OPBUFF_ADDR	31:0	Address to store Operation Buffer in system memory	OPBUFF address

10.7.38 SCA Modulo Data Register (SCA_MODDATA, Offset 0x0130)

Table 10-38. SCA_MODDATA (Offset 0x0130)

Position	31	30	29	28	27	26	25	24
Field	MODDATA[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	MODDATA [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	MODDATA [15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	MODDATA [7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
MODDATA	31:0	This register is used to load modulo to internal register. The internal modulo register is 256-bit width, it is filled by setting SCA_CN.MODADDR bits (offset of the 32-bit data) than the corresponding data must be written in MODDATA,	

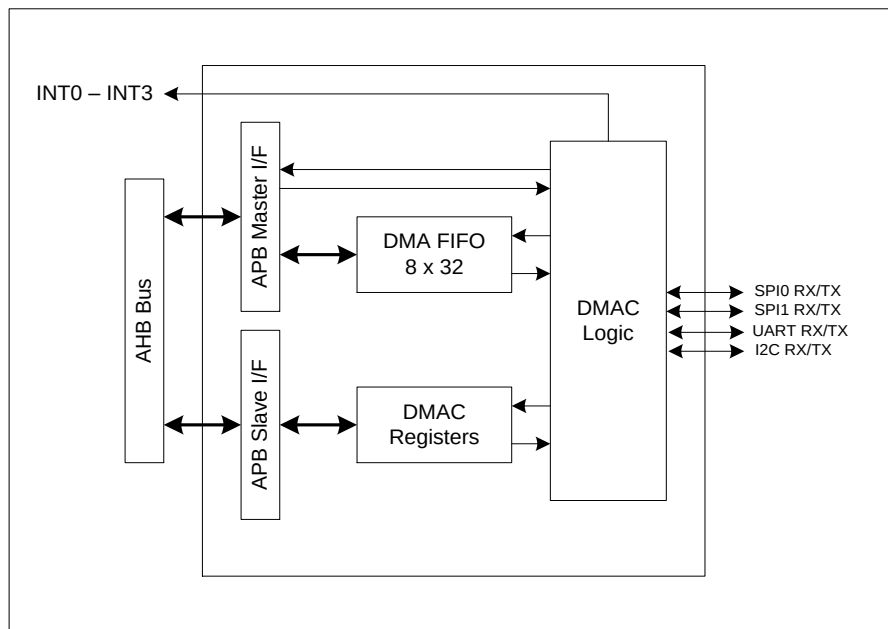
11.0 DMA Controller

The DMA Controller (DMAC) provides fully programmable, chaining capable DMA channels that support the following categories of transfer:

- 4-channel
- Peripheral-to-Memory
- Memory-to-Peripheral
- Memory-to-Memory

All DMA transactions consist of an AHB burst read into the DMA FIFO followed immediately by an AHB burst write from the FIFO.

Figure 11-1. DMA Controller Block Diagram



11.1 DMA Channel Operation

Each channel of the DMA channel is governed by a set of registers:

- Channel Configuration Register
- Channel Status Register
- Channel Source Register
- Channel Destination Register
- Channel Count Register

Additionally, each channel has a set of reload registers which allows for easy chaining of DMA buffers on count-to-zero (CTZ) condition:

- Channel Source Reload Register
- Channel Destination Reload Register
- Channel Count Reload Register

Using these registers allows each channel the following features:

- Full 32-bit source and destination addresses with 24-bit (16M-byte) address increment capability
- Up to 16Mbytes for each DMA buffer

- Programmable burst size
- Programmable priority
- Interrupt upon count-to-zero.
- Abort on error

11.2 DMA Channel Arbitration and DMA Bursts

The DMA Controller contains an internal arbiter to allow any enabled channels to access the AHB and move data. A DMA channel must be enabled before it can request from the arbiter. The Channel Enable (DMA_CFG.CHEN) bit is used to enable a channel.

When disabling a channel, the programmer must poll Channel Status (DMA_ST.CH_ST) to determine if the channel is truly disabled. In general, CH_ST follows the setting of the DMA_CFG.CHEN. However, it is automatically cleared under the following conditions:

- Bus error (cleared immediately)
- Count-to-zero with RLDEN=0 (cleared at the end of the AHB R/W burst).
- CHEN cleared by programmer (cleared at the end of the AHB R/W burst).

Whenever CH_ST automatically transitions from 1 to 0, the corresponding DMA_CFG.CHEN bit is also cleared (if not cleared already). During an AHB read/write burst, attempting to disable an active channel is delayed until burst completion.

Once a channel has been programmed and enabled, it generates a request to the arbiter either immediately (for Memory-to-Memory DMA) or whenever its associated peripheral requests DMA (for Memory-to-Peripheral or Peripheral-to-Memory DMA).

The arbiter grants requests to a single channel at a time. Granting is done first based on priority – a higher priority request is always granted. Within a given priority level, requests are granted on a round-robin basis.

Once a channel's request is granted, it executes a DMA access in two steps:

1. Burst movement of data from the source into the DMA FIFO.
2. Burst movement of data from the DMA FIFO to the destination.

Any required data alignment is achieved during when moving the data in and out of the DMA FIFO.

Once granted, only an error condition interrupts the execution of the two steps listed above. Clearing the enable bit or the occurrence of a higher priority request does not interrupt an on-going DMA transfer.

Once the two steps are complete, the channel relinquishes its grant.

Request Select (DMA_CFG.REQSEL) determines which request is used to initiate a DMA burst. In the case of memory-to-memory DMA, the channel is treated as always requesting DMA access. The Priority (PRI) bits set the DMA channel priority.

11.3 DMA Source and Destination Addressing

For memory addresses, the DMA_SRC and DMA_DST fields are used to program the addresses of the source and destination. For peripherals, however, all or part of the address is fixed based on the settings of the Request Select Field. (DMA_CFG.REQSEL). This is done to prevent data movement to/from secure parts of the device.

Table 11-1. Source and Destination Address Definition shows how the source and destination addresses as well as the address increment controls are constructed based on the "REQ SEL" setting.

"P" in the "SRCINC" or "DSTINC" field indicates that the field is programmable and set according to the bit field in the DMA_CFG. If there is a zero in the column, the field will be forced to zero.

Table 11-1. Source and Destination Address Definition

Request Select	Transfer	Source Address	SRC INC	Destination Address	DST INC
000000 (0x00)	Mem-to-Mem	DMA_SRC	P	DMA_DST	P
000001 (0x01)	SPI0 RX	SPI0 Data Register 0x4004_6000	0	DMA_DST	P
000010 (0x02)	SPI1 RX	SPI1 Data Register 0x4004_7000	0	DMA_DST	P
000011 (0x03) to 000110 (0x06)	<i>Reserved</i>				
000111 (0x07)	I2C 0 RX	I2C Data Register 0x4001_D02C	0	DMA_DST	P
001000 (0x08) to 011011 (0x1B)	<i>Reserved</i>				
011100 (0x1C)	UART0 RX	UART0 Data Register 0x4002_0020	0	DMA_DST	P
011011 (0x1D) to 100000 (0x20)	<i>Reserved</i>				
100001 (0x21)	SPI0 TX	DMA_SRC	P	SPI0 Data Register 0x4004_6000	0
100010 (0x22)	SPI1 TX	DMA_SRC	P	SPI1 Data Register 0x4004_7000	0
100011 (0x23) to 100110 (0x26)	<i>Reserved</i>				
100111 (0x27)	I2C 0 TX	DMA_SRC	P	I2C Data Register 0x4001_D02C	0
101000 (0x28)) to 111011 (0x3B)	<i>Reserved</i>				
111100 (0x3C)	UART0 TX	DMA_SRC	P	UART0 Data Register 0x4002_0020	0
111101 (0x3D) to 111111 (0x3F)	<i>Reserved</i>				

11.4 Data Movement from Source to DMA FIFO

There are several registers to control the movement of data into the DMA FIFO. The source can be either a peripheral or memory.

DMA_SRC: Source address. If the increment enable is set, this increments on every read cycle of the burst.

DMA_CNT: Number of bytes to transfer before a “count-to-zero” condition occurs. This register is decremented on each read of the burst.

BRST: Burst Size (1-32). This determines the maximum number of bytes to be moved during the burst read. For the final burst, the actual number moved is never be more than the current value of DMA_CNT.

SRCWD: Source Width. This determines the maximum data width to be used during each read of the AHB burst (byte, half-word, or word). The actual AHB width may be less if DMA_CNT is not great enough to supply all the needed bytes.

SRCINC: Source Increment Enable. This enables the incrementing of DMA_SRC. See Table 11-1 for restriction.

Table 11-2 depicts how data is moved into the DMA FIFO based on the settings of the DMA_SRC[1:0] and the “Source Width” bits. In cases where the width of the device is larger than the current value of DMA_CNT, the DMA Controller performs an AHB cycle corresponding to a smaller width.

Table 11-2. Inbound Data Alignment

Source Device Characteristics					Register Field Values				Resulting AHB Burst
Data Word Size	Active Byte Lanes				SRC Width		DMA_SRC[1:0]		
	3	2	1	0	1	0	0	0	
8				X	0	0	0	0	One AHB byte read for each byte moved. Only data from the indicated byte lane is moved into the DMA FIFO (LSByte first).
			X				0	1	
		X					1	0	
	X						1	1	
16			X	X	0	1	0	0	One AHB half-word read for each half-word moved. Only data from the indicated byte lanes is moved into the DMA FIFO (least significant half-word first).
	X	X					1	0	
32	X	X	X	X	1	0	0	0	One AHB word read for each word moved. The entire word is moved into the DMA FIFO.

11.5 Data Movement from the DMA FIFO to Destination

There are several registers to control the burst movement of data out of the DMA FIFO. The destination can be either a peripheral or memory.

DMA_DST: Destination Address. If the increment enable is set, this increments on every write cycle of the burst.

DMA FIFO: Counter (Internal and not accessible). THIS IS AN INTERNAL REGISTER WHICH CANNOT BE ACCESSED. This indicates the number of bytes currently held within the DMA FIFO. This register is automatically incremented on AHB reads and decremented on AHB writes.

BRST: Burst Size (1-32). This determines the maximum number of bytes to be moved during a single AHB read/write burst. The actual number moved is never more than the current value of DMA FIFO Counter.

DSTWD: Destination Width. This determines the maximum data width to be used during each write of the AHB burst (byte, half-word, or word). The actual width never represents a greater number of bytes than the current value of the DMA FIFO Counter.

DSTINC: Destination Increment Enable. This enables the incrementing of DMA_DST. See Table 11-1 for restriction.

Table 11-3 depicts how data is moved out of the DMA FIFO based on the settings of DMA FIFO COUNTER, DMA_DST[1:0] and the “DST WIDTH”. In cases where the width of the device is larger than the number of bytes left in the DMA FIFO, the DMA Controller performs an AHB cycle corresponding to a smaller width.

Table 11-3. Outbound Data Alignment

Destination Device Characteristics					Register Field Values				Resulting AHB Burst
Data Word Size	Active Byte Lanes				DST Width		DMA_DST[1:0]		
	3	2	1	0	1	0	0	0	
8				X	0	0	0	0	One AHB byte write for each byte moved. Bytes are moved out of the DMA FIFO (LSByte first) onto the indicated byte lane.
			X				0	1	
		X					1	0	
	X						1	1	
16			X	X	0	1	0	0	One AHB half-word write for each half-word moved. Half-words are moved out of the DMA FIFO (least significant half-word first) onto the indicated byte lanes.
	X	X					1	0	
32	X	X	X	X	1	0	0	0	One AHB word write for each word moved. The entire word is moved out of the DMA FIFO.

11.6 Memory Buffer Alignment

The DMA dynamically adjusts the transfer size to provide correct buffer alignment. There are no restrictions on the buffer alignment.

11.7 Count-to-Zero Condition

When an AHB channel burst complete, the DMAC checks to see if DMA_CNT is decremented to 0. If so, a count-to-zero (CTZ) condition occurs.

Upon count-to-zero, depending on RLDEN, there are two possible responses:

1. RLDEN=1. DMA_SRC, DMA_DST, and DMA_CNT is loaded from the reload registers. The channel remains active and continues operating using the newly loaded address/count values and the previously programmed configuration values.
2. RLDEN=0. The channel is disabled and DMA_ST.CH_ST is cleared to 0.

11.8 Chaining Buffers

The reload registers may be used for chaining buffers together. This eases the DMA ISR response time requirements and allows the DMA to continue to service a request without immediate processing from the CPU.

The Reload Enable Bit (RLDEN=1) automates a reload of DMA_SRC, DMA_DST, and DMA_CNT upon a count-to-zero condition. The channel remains active. The DMA_SRC, DMA_DST, and DMA_CNT are loaded with values from the reload registers. The DMA operation continues with the new DMA buffer. The RLDEN bit can be set either in the DMA_CFG or DMA_CNT_RLD registers.

To configure a channel for buffer chaining both the base and reload DMA registers must be initialized prior to starting the DMA. These are the DMA_CFG, DMA_SRC, DMA_DST, DMA_CNT, DMA_SRC_RLD, DMA_DST_RLD and DMA_CNT_RLD. When the DMA_CNT_RLD register is written, the RLDEN bit must not be set. Any writes to the DMA_CFG register prior to completion of this initialization must not set the CHEN and RLDEN bits. After all registers are initialized, the last operation in the initialization is a write to the DMA_CFG register which sets both the CHEN and RLDEN bits. This starts the DMA. Subsequent DMA interrupts are serviced by writing only to the reload registers, setting DMA_CNT_RLD.RLDEN.

The CTZIEN bit must be set in the DMA_CFG register to receive an interrupt after each buffer is DMA'd. The Channel Disable Interrupt Enable bit should also be set to provide an interrupt in the case of a bus error.

Caution: Setting the CHEN via a write to the DMA_CFG register and separately setting the RLDEN bit via a write to the DMA_RLD_CNT register risks a race condition between a DMA completion interrupt service routine initializing the reload registers (for the third buffer) before the software initialization of these registers for the second buffer.

When the first DMA transfer completes (based on DMA_CNT value), a count to zero interrupt occurs. The DMA_SRC, DMA_DST and DMA_CNT registers are reloaded from the corresponding reload registers. The DMA_ST register indicates both “reload” and “count to zero” events occurred. The Channel Status bit (CH_ST) is typically a 1, indicating the DMA is now busy with the second DMA transfer defined in the reload registers. If CH_ST=0, this indicates both the initial and second DMA transfers have completed. If there are additional buffers to chain, the interrupt service routine initializes the DMA_SRC_RLD, DMA_DST_RLD, and DMA_CNT_RLD registers, setting DMA_CNT_RLD.RLDEN bit as the last operation. The interrupt service routine does not write to the DMA_CFG, DMA_SRC, DMA_DST, and DMA_CNT registers, just the reload registers.

To prevent improper operation, program the address bits before setting the RLDEN bit.

11.9 DMA Interrupts

Interrupts from each channel can be individually enabled by setting the corresponding interrupt enable bits (DMA_CN.CHIEN). When an interrupt is pending, the corresponding Channel Interrupt Pending (DMA_INT.IPEND) bit is set to 1. This is a read-only status. The interrupt must be cleared in the DMA_ST.

A channel interrupt (DMA_ST.IPEND=1) can be caused by:

- Count-to-zero interrupt (DMA_CFG.CTZIEN=1). If enabled, all count-to-zero occurrences set IPEND.
- Channel Disable interrupt (DMA_CFG.CHDIEN=1). If enabled, any clearing of CH_ST sets PEND. The DMA_ST can be examined to see which reason(s) caused the disable. The CHDIEN also enables the Time-out interrupt. The Time-out interrupt does not clear the CH_ST bit

To clear the channel interrupt, write 1 to the cause of the interrupt (CTZ_ST, RLD_ST, BUS_ERR, TO_ST).

When running in normal mode without buffer chaining (RLDEN = 0), it is recommended to set only the DMA_CFG.CHDIEN. An interrupt is generated upon DMA completion or an error condition (bus error or time-out error).

When running in buffer chaining mode (RLDEN=1), it is recommended to set both the CHDIEN and CTZIEN bits. The count-to-zero interrupts occur on completion of each DMA (count reaches zero and reload occurs). The setting of CHDIEN insures that an error condition also generates an interrupt. If the CTZIEN=0, the only DMA completion interrupt will be when the DMA completes and RLDEN=0 (final DMA).

11.10 Channel Timeouts

Each channel can be configured to generate an interrupt when its associated request line is inactive for a given period. An example use of this feature is to determine an idle UART receive channel. The timeout mechanism consists of a single global 24-bit prescaler and per-channel programmable 8-bit timer/counters.

The Channel Disable Interrupt Enable bit must be set for the Timeout to generate an interrupt.

11.10.1 Global 24-bit Prescaler

A single global 24-bit clock prescaler is used to divide hclk. There are three taps provided from this divider:

- hclk/256
- hclk/64k
- hclk/16M

11.10.2 8-bit Timer/Counter

Each channel has an individual 8-bit timer/counter. The Prescale Select (DMA_CFG.PSSEL) bits select the prescaler taps used as the timer clock source.

The Timeout Select (DMA_CFG.TOSEL) bits set the limit at which the 8-bit timer/counter should count before generating an interrupt.

The 8-bit timer/counter is reset whenever any of the following conditions occurs:

1. The DMA request line programmed for the channel is activated.
2. The channel is disabled for any reason (DMA_ST.CH_ST=0).

Any of the 8-bit timer can be disabled by setting the PSSEL bits to 00b. When all channels of the DMAC are disabled, the global prescaler is also disabled.

Normally, the 8-bit timer starts counting as soon as the channel is enabled and the PSSEL bits are non-zero. But if Request Wait is set (DMA_CFG.REQWAIT=1), the timer starts counting only after the first DMA request is received from the peripheral.

The timeout period can be calculated using the following equation:

$$T_{\text{TIMEOUT}} = T_{\text{HCLK}} * N_{\text{PSSEL}} * N_{\text{TOSEL}}$$

For an hclk frequency of 90MHz, and PSSEL=10b and TOSEL=100b, the calculation is:

$$T_{\text{TIMEOUT}} = (1/90\text{MHz}) * 65536 * 32 = 23.3\text{ms}$$

11.11 Channel and Register Access Restrictions

The programmer is free to write to any registers while a channel is disabled, but there are certain restrictions when a channel is enabled. The Channel Status bit indicates if the channel is enabled or not.

Because an active channel may be in the middle of an AHB read/write burst, the user should not write to DMA_SRC, DMA_DST, or DMA_CNT while a channel is active (CH_ST=1).

The programmer can disable any DMA channel by clearing CHEN. The user must then poll CH_ST to verify that the channel is disabled. When clearing Channel Enable bit, the user should perform a read-modify-write to ensure that other bits of the DMA_CFG are not modified.

CAUTION: When disabling a DMA channel, ensure that system interrupts are configured appropriately.

11.12 Memory-to-Memory DMA

Memory-to-memory transfers are performed as if the request is always active. This means that the DMA channel generates an almost constant request for the bus until its transfer is complete. So, assign a lower priority to channels executing memory-to-memory transfers to prevent starvation of other DMA channels.

11.13 DMA Registers

Address assignments for the DMA registers are outlined in Table 11-4 and Table 11-5.

Table 11-4. DMA Register Address (Base ADDR = 0x4002_8000)

Offset	Access	Register	Description
0x0000	RW	DMA_CN	DMA Control Register
0x0004	RW	DMA_INT	DMA Interrupt Status Register
0x0100	RW	DMA0	DMA Channel 0 Base Address
0x0120	RW	DMA1	DMA Channel 1 Base Address
0x0140	RW	DMA2	DMA Channel 2 Base Address
0x0160	RW	DMA3	DMA Channel 3 Base Address

Each channel has a set of associated registers. The relative offsets for these registers with respect to the channel base address are outlined in Table 11-5.

Table 11-5. DMA Channel Registers Offset

Offset	Access	Register	Description
0x0000	RW	DMAx_CFG	DMA Channel x Configuration Register
0x0004	RW	DMAx_ST	DMA Channel x Status Register
0x0008	RW	DMAx_SRC	DMA Channel x Source Register
0x000C	RW	DMAx_DST	DMA Channel x Destination Register
0x0010	RW	DMAx_CNT	DMA Channel x Count Register
0x0014	RW	DMAx_SRC_RLD	DMA Channel x Source Reload Register
0x0018	RW	DMAx_DST_RLD	DMA Channel x Destination Reload Register
0x001C	RW	DMAx_CNT_RLD	DMA Channel x Count Reload Register

Since the registers are identical for all the channels, only registers associated with DMA Channel 0 is described in this document as the same definition applies to all channels.

All DMA channels operate identically. only Channel 0 registers are described. For all other channel operations, refer to Channel 0 registers description.

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

11.13.1 DMA Control Register (DMA_CN, Offset 0x0000)

Table 11-6. DMA_CN (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[3:0]				CHIEN[3:0]			
Reset	0000				0000			
Access	R				RW			

Name	Bits	Description	Settings
CHIEN	3:0	Channel Interrupt Enable. Each bit enables the corresponding channel interrupt.	0: Disable 1: Enable
RFU	31:4	Reserved	N/A

11.13.2 DMA Interrupt Register (DMA_INT, Offset 0x0004)

Table 11-7. DMA_INT (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[3:0]				IPEND[3:0]			
Reset	0000				0000			
Access	R				R			

Name	Bits	Description	Settings
IPEND	3:0	Channel Interrupt. Each bit represents interrupt from the corresponding channel. To clear an interrupt, all active interrupt bits of the DMA_ST must be cleared. The interrupt bits are set only if their corresponding interrupt enable bits are set in DMA_CN.	0: No Interrupt 1: Interrupt Pending
RFU	31:4	Reserved	N/A

11.13.3 DMA Channel 0 Configuration Register (DMA0_CFG, Offset 0x0100)

Table 11-8. DMA0_CFG (Offset 0x0100)

Position	31	30	29	28	27	26	25	24
Field	CTZIEN	CHDIEN	RFU	BRST [4:0]				
Reset	0	0	0	0 0000				
Access	RW	RW	R	RW				
Position	23	22	21	20	19	18	17	16
Field	RFU	DSTINC	DSTWD[1:0]		RFU	SRCINC	SRCWD[1:0]	
Reset	0	0	00		0	0	00	
Access	R	RW	RW		R	RW	RW	
Position	15	14	13	12	11	10	9	8
Field	PSSEL[1:0]		TOSEL[2:0]			REQWAIT	REQSEL[5:4]	
Reset	00		000			0	00	
Access	RW		RW			RW	RW	
Position	7	6	5	4	3	2	1	0
Field	REQSEL[3:0]				PRI[1:0]		RLDEN	CHEN
Reset	0000				00		0	0
Access	RW				RW		RW	RW

Name	Bits	Description	Settings
CHEN	0	Channel Enable. This bit is automatically cleared when DMA_ST.CH_ST changes from 1 to 0.	0: Disable this channel. 1: Enable this channel.
RLDEN	1	Reload Enable. Setting this bit to 1 enables DMA_SRC, DMA_DST and DMA_CNT to be reloaded with their corresponding reload registers upon count-to-zero. This bit is also writeable in the Count Reload Register. Refer to the description on Buffer Chaining for use of this bit. If buffer chaining is not used this bit must be written with a 0. This bit should be set after the reload registers have been programmed.	0: Disabled 1: Enabled.
PRI	3:2	DMA Priority.	00: Highest Priority 11: Lowest Priority
REQSEL	9:4	Request Select. Select DMA request line for this channel. If memory-to-memory is selected, the channel operates as if the request is always active. Refer to Table 11-1 for the decode for this field.	
REQWAIT	10	Request Wait Enable. When enabled, delay timer starts until DMA request transitions from active to inactive.	0: Start timer normally. 1: Delay timer start.
TOSEL	13:11	Time-Out Select. Selects the number of prescale clocks seen by the channel timer before a timeout condition is generated for this channel. Important note: Since the prescaler runs independent of the individual channel timers, the actual number of Pre-Scale clock edges seen has a margin of error equal to a single Pre-Scale clock. This is reflected in the settings column (e.g. 3-4, not just 4).	000: 3-4 001: 7-8 010: 15-16 011: 31-32 100: 63-64 101: 127-128 110: 255-256 111: 511-512
PSSEL	15:14	Pre-Scale Select. Selects the Pre-Scale divider for timer clock input	00: Disable timer 01: hclk / 256 10: hclk / 64k 11: hclk / 16M

Name	Bits	Description	Settings
SRCWD	17:16	Source Width. In most cases, this is the data width of each AHB transactions. However, the width is reduced in the cases where DMA_CNT indicates a smaller value.	00: Byte 01: Half-word 10: Word 11: Reserved (Byte width if set)
SRCINC	18	Source Increment Enable. This bit enables DMA_SRC increment upon every AHB transaction. This bit is forced to 0 for DMA receive from peripherals.	0: Increment disabled. 1: Increment enabled.
RFU	19	Reserved.	N/A
DSTWD	21:20	Destination Width. Indicates the width of the each AHB transactions to the destination peripheral or memory. (The actual width may be less than this if there are insufficient bytes in the DMA FIFO for the full width).	00: Byte 01: Half-word 10: Word 11: Reserved (Byte width if set)
DSTINC	22	Destination Increment Enable. This bit enables DMA_DST increment upon every AHB transaction. This bit is forced to 0 for DMA transmit to peripherals.	0: Increment disabled. 1: Increment enabled.
RFU	23	Reserved.	N/A
BRST	28:24	Burst Size. The number of bytes to be transferred into and out of the DMA FIFO in a single burst.	00000: 1 byte 00001: 2 bytes 00010: 3 bytes ... 11111: 32 bytes
RFU	29	Reserved.	N/A
CHDIEN	30	Channel Disable Interrupt Enable. When enabled, the IPEND is set to 1 whenever CH_ST changes from 1 to 0.	0: Interrupt disabled. 1: Interrupt enabled.
CTZIEN	31	Count-to-zero Interrupts Enable. When enabled, the IPEND is set to 1 whenever a count-to-zero event occurs.	0: Interrupt disabled. 1: Interrupt enabled.

11.13.4 DMA Channel 0 Status Register (DMA0_ST, Offset 0x0104)

Table 11-9. DMA0_ST (Offset 0x0104)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU	TO_ST	RFU	BUS_ERR	RLD_ST	CTZ_ST	IPEND	CH_ST
Reset	0	0	0	0	0	0	0	0
Access	R	RW1C	R	RW1C	RW1C	RW1C	R	R

Name	Bits	Description	Settings
CH_ST	0	Channel Status. This bit is used to indicate to the programmer when it is safe to change the configuration, address, and count registers for the channel. Whenever this bit is cleared by hardware, the DMA_CFG.CHEN bit is also cleared (if not cleared already).	0: Channel disabled. 1: Channel enabled.
IPEND	1	Channel Interrupt.	0: No interrupt 1: Interrupt pending
CTZ_ST	2	Count-to-Zero (CTZ) Status.	Read: 0: CTZ has not occurred. 1: CTZ has occurred. Write: 0: No effect 1: Write 1 to clear
RLD_ST	3	Reload Status.	Read: 0: Reload has not occurred. 1: Reload has occurred. Write: 0: No effect 1: Write 1 to clear
BUS_ERR	4	Bus Error. Indicates that an AHB abort is received and the channel is disabled.	Read: 0: No error found 1: An AHB bus error occurred Write: 0: No effect 1: Write 1 to clear
RFU	5	Reserved	N/A

Name	Bits	Description	Settings
TO_ST	6	Timeout Status.	Read: 0: No timeout 1: A timeout has occurred Write: 0: No effect 1: Write 1 to clear
RFU	31:7	Reserved	N/A

11.13.5 DMA Channel 0 Source Register (DMA0_SRC, Offset 0x0108)

Table 11-10. DMA0_SRC (Offset 0x0108)

Position	31	30	29	28	27	26	25	24
Field	SRC[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	SRC[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	SRC[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	SRC[7:0]							
Reset	000 0000							
Access	RW							

Name	Bits	Description	Settings
SRC	31:0	Source Device Address. If SRCINC=1, the counter bits are incremented by 1,2, or 4, depending on the data width of each AHB cycle. For peripheral transfers, some or all the actual address bits are fixed (see Table 11-1). If SRCINC=0, this register remains constant. In the case where a count-to-zero condition occurs while RLDEN=1, the register is reloaded with the contents of DMA_SRC_RLD.	

11.13.6 DMA Channel 0 Destination Register (DMA0_DST, Offset 0x010C)

Table 11-11. DMA0_DST (Offset 0x010C)

Position	31	30	29	28	27	26	25	24
Field	DST[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DST[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DST[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DST[7:0]							
Reset	000 0000							
Access	RW							

Name	Bits	Description	Settings
DST	31:0	Destination Device Address. For peripheral transfers, some or all the actual address bits are fixed (see Table 11-1). If DSTINC=1, this register is incremented on every AHB write out of the DMA FIFO. They are incremented by 1,2, or 4, depending on the data width of each AHB cycle. In the case where a count-to-zero condition occurs while RLDEN=1, the register is reloaded with DMA_DST_RLD.	

11.13.7 DMA Channel 0 Count Register (DMA0_CNT, Offset 0x0110)

Table 11-12. DMA0_CNT (Offset 0x0110)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	CNT[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	CNT[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	CNT[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
CNT	23:0	DMA Counter. The user loads this register with the number of bytes to transfer. This counter decreases on every AHB cycle into the DMA FIFO. The decrement is 1, 2, or 4 depending on the data width of each AHB cycle. When the counter reaches '0', a count-to-zero condition is triggered.	0x000000: 0 Byte 0x000001: 1 Byte 0x000002: 2 Bytes ... 0xFFFFF: 16,777,215 Bytes
RFU	31:24	Reserved	N/A

11.13.8 DMA Channel 0 Source Reload Register (DMA0_SRC_RLD, Offset 0x0114)

Table 11-13. DMA0_SRC_RLD (Offset 0x0114)

Position	31	30	29	28	27	26	25	24
Field	RFU	SRC_RLD[30:24]						
Reset	0	000 0000						
Access	R	RW						
Position	23	22	21	20	19	18	17	16
Field	SRC_RLD[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	SRC_RLD[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	SRC_RLD[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
SRC_RLD	30:0	Source Address Reload Value. The value of this register is loaded into DMA0_SRC upon a count-to-zero condition.	
RFU	31	Reserved	N/A

11.13.9 DMA Channel 0 Destination Reload Register (DMA0_DST_RLD, Offset 0x0118)

Table 11-14. DMA0_DST_RLD (Offset 0x0118)

Position	31	30	29	28	27	26	25	24
Field	DST_RLD[30:24]							
Reset	000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DST_RLD[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DST_RLD[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DST_RLD[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DST_RLD	30:0	Destination Address Reload Value. The value of this register is loaded into DMA0_DST upon a count-to-zero condition.	
RFU	31	Reserved	N/A

11.13.10 DMA Channel 0 Count Reload Register (DMA0_CNT_RLD, Offset 0x011C)

Table 11-15. DMA0_CNT_RLD (Offset 0x011C)

Position	31	30	29	28	27	26	25	24
Field	RLDEN	RFU[6:0]						
Reset	0	000 0000						
Access	RW	R						
Position	23	22	21	20	19	18	17	16
Field	CNT_RLD[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	CNT_RLD[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	CNT_RLD[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
CNT_RLD	23:0	Count Reload Value. The value of this register is loaded into DMA0_CNT upon a count-to-zero condition.	
RFU	30:24	Reserved	N/A
RLDEN	31	Reload Enable. This bit should be set after the address reload registers are programmed. This bit is automatically cleared to 0 when reload occurs.	0: Reload disabled. 1: Reload enabled.

12.0 Flash Controller

The MAX32520 has 2MB of PUF encrypted flash memory with low latency on-the-fly decryption of flash execution. This process is handled transparently by the Memory Protection Unit (MPU). The address span of the Flash memory is shown in Table 2-1. The flash memory has the following features:

- Up to 2MB total internal flash memory
- 128 pages
- 16,384 bytes per page
- 2,048 words by 128 bits per page
- 128-bit data reads and writes
- Page erase and mass erase support
- Write protection

The internal flash memory is organized in two equal size banks, which can be used as a single contiguous 2MB memory or two 1MB memories, each for storing user application and data. This feature allows reliable and failsafe remote firmware updates. The internal flash memory is programmable via JTAG interface (in-system) or directly with user application code (in-application).

The flash is organized as an array of pages. Each page is 2,048 words by 128 bits, or 16,384 bytes per page. Table 12-1 shows the start address and end address for each flash instance. The internal flash memory is mapped with a start address of 0x1000 0000 and an end address of 0x101F FFFF for a total of 2MB.

Table 12-1. Flash Memory Configuration

Instance Number	Page Number	Size	Start Address	End Address
Bank0	1	16,384 Bytes	0x1000 0000	0x1000 3FFF
	2	16,384 Bytes	0x1000 4000	0x1000 7FFF
	3	16,384 Bytes	0x1000 8000	0x1000 BFFF
	4	16,384 Bytes	0x1000 C000	0x1000 FFFF
	.	.	.	.
	63	.	0x100F 8000	0x100F BFFF
Bank1	64	.	0x100F C000	0x100F FFFF
	1	.	0x1010 0000	0x1010 3FFF
	2	.	0x1010 4000	0x1010 7FFF
	3	.	0x1010 8000	0x1010 BFFF
	4	.	0x1010 C000	0x1010 FFFF
	.	.	.	.
	63	16,384 Bytes	0x101F 8000	0x101F BFFF
	64	16,384 Bytes	0x101F C000	0x101F FFFF

12.1 Flash Memory Operations

12.1.1 Clock Configuration

The Flash Controller requires a 1MHz peripheral clock for operation. The input clock to the Flash Controller block is the system clock, SYSCLK. Use the Flash Controller clock divisor to generate a 1MHz, as shown in the following equation. For the 120MHz Oscillator as the system clock, the FLSH_CLKDIV.CLKDIV should be set to 120 (0x78).

$$\text{FLSH\_CLKDIV.CLKDIV} = f_{\text{SYSTEMCLOCK}}/1\text{MHz}$$

12.1.2 Lock Protection

A locking mechanism prevents accidental memory writes and erases. All writes and erase operations require the `FLSH_CN.UNLOCK` field set to 0x2 prior to starting the operation. Writing any other value to this field, `FLSH_CN.UNLOCK`, results in:

- 1) The flash instance remaining locked.
- 2) The flash instance becoming locked from the unlocked state.

Note: If a write, page erase or mass erase operation is started and the unlock code was not set to 0x2, the flash controller hardware sets the access fail flag, `FLSH_INT.AF`, to indicate an access violation occurred.

12.1.3 Flash Write

Writes to a flash location are only successful if the targeted location is already in its erased state. Perform the following steps to write to a flash memory instance:

1. If desired, enable flash controller interrupts by setting the `FLSH_INT.AFIE` and `FLSH_INT.DONEIE` bits.
2. Read the `FLSH_CN.PEND` bit until it returns 0.
3. Configure `FLSH_CN.CLKDIV` to match the System Clock frequency.
4. Set the `FLSH_ADDR[31:4]` register to target address.
5. Set `FLSH_DATA3`, `FLSH_DATA2`, `FLSH_DATA1`, and `FLSH_DATA0` to the data to write.
`FLSH_DATA3` is the most significant word and `FLSH_DATA0` is the least significant word. Each word of the data to write follows the little-endian format where the least significant byte of the word is stored at the lowest-numbered byte and the most significant byte is stored at the highest-numbered byte.
6. Set `FLSH_CN.UNLOCK` to 0x2 to unlock the flash instance.
7. Set `FLSH_CN.WR` to 1. This field is automatically cleared by the Flash Controller when the write operation is finished.
8. `FLSH_INT.DONE` is set by hardware when the write completes and if an error occurred. The `FLSH_INT.AF` is set. These bits generate a flash IRQ if the interrupt enable bits are set.
9. Set `FLSH_CN.UNLOCK` to any value other than 0x2 to relock the flash instance.

12.1.4 Flash Page Erase

Flash page erase address is written to the Flash Address Register (`FLSH_ADDR`). Only `FLSH_ADDR[20:14]` is valid. The application then sets the Page Erase bit (`PGE`) to initiate the erase. The `PGE` bit clears itself when the operation completes.

Perform the following to erase a page of a flash memory instance:

1. If desired, enable flash controller interrupts by setting the `FLSH_INT.AF` and `FLSH_INT.DONE` bits.
2. Read the `FLSH_CN.PEND` bit until it returns 0.
3. Configure `FLSH_CLKDIV.CLKDIV` to match the System Clock frequency.
4. Set the `FLSH_ADDR` register to an address within the target page to be erased. `FLSH_ADDR[14:0]` are ignored by the Flash Controller to ensure the address is page aligned.
5. Set `FLSH_CN.UNLOCK` to 0x2 to unlock the flash instance.
6. Set `FLSH_CN.ERASE_CODE` to 0x55 for page erase.
7. Set `FLSH_CN.PGE` to 1 to start the page erase operation.
8. The `FLSH_CN.PEND` bit is set by the flash controller while the page erase is in progress and the `FLSH_CTRL.PGE` and `FLSH_CTRL.PEND` are cleared by the flash controller when the page erase is complete.
9. `FLSH_INT.DONE` is set by the hardware when the page erase completes and if an error occurred, the `FLSH_INT.AF` flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
10. Set `FLSH_CN.UNLOCK` to any value other than 0x2 to re-lock the flash instance.

12.1.5 Flash Mass Erase

Mass erase clears the internal flash memory on an instance basis. Perform the following steps to mass erase a single flash memory instance:

1. Read the `FLSH_CN.PEND` bit until it returns 0.

2. Configure FLSH_CLKDIV.CLKDIV to match the System Clock frequency.
3. Set FLSH_CN.UNLOCK to 0x2 to unlock the internal flash.
4. Set FLSH_CN.ERASE_CODE to 0xAA for mass erase.
5. Set FLSH_CN.ME to 1 to start the mass erase operation.
6. The FLSH_CN.PEND bit is set by the flash controller while the mass erase is in progress and the FLSH_CN.ME and FLSH_CN.PEND are cleared by the flash controller when the mass erase is complete.
7. FLSH_INT.DONE is set by the flash controller when the mass erase completes and if an error occurred, the FLSH_INT.AF flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
8. Set FLSH_CN.UNLOCK to any value other than 0x2 to re-lock the flash instance.

12.1.6 Register Access Restriction

Write to any of the Flash registers are ignored when the Flash is busy as indicated by any of the PGE, ME, or WR bits being set, only one of which can be set at any one time.

12.1.7 Interrupt

The Flash Done Interrupt (FLSH_INT.DONE) bit is set at the end of a Flash operation. The DONE bit is set whether the operation succeeds or not. In the case of an error, the Access Fail bit (FLSH_CN.AF) bit is also set to 1.

12.1.8 Flash Decryption

This device has an on-the-fly decryption engine. This engine decrypts the flash content and executes it on-the-fly. Any code and data write to the decrypted area of the flash is separately encrypted prior to writing it to the flash. The encryption/decryption key is PUF_KEY1. This key should be used along with the Cryptographic AES engine to encrypt the data/code before writing it to flash. This key can be selected by writing the appropriate value to the CIPHER_CTRL.SRC.

12.1.9 Flash Encrypted Region

The *non-encrypted* region of the flash is between the FLASH_SBA and FLASH_STA registers in the SFE registers. The addresses below FLASH_SBA and above FLASH_STA are encrypted. Refer to the 20.0 Serial Flash Emulation Interface for more information on these registers.

12.2 Flash Controller Registers

Address assignments for the Flash Controller Registers are outlined in Table 12-2.

Table 12-2. Flash Controller Register Address (Base ADDR = 0x4002_9000)

Offset	Access	Register	Description
0x0000	RW	FLSH_ADDR	Flash Address Register
0x0004	RW	FLSH_CLKDIV	Flash Clock Divide Register
0x0008	RW	FLSH_CN	Flash Control Register
0x000C ... 0x0020	RW	RFU	
0x0024	R	FLSH_INT	Flash Interrupt Register
0x0028 ... 0x002C	R	RFU	
0x0030	RW	FLSH_DATA0	Flash Data Register 0
0x0034	RW	FLSH_DATA1	Flash Data Register 1
0x0038	RW	FLSH_DATA2	Flash Data Register 2
0x003C	RW	FLSH_DATA3	Flash Data Register 3
0x0040	RW	ACNTL	Access Control Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

12.2.1 Flash Address Register (FLSH_ADDR, Offset 0x0000)

Table 12-3. FLSH_ADDR (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	ADDR[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	ADDR[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	ADDR[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	ADDR[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
ADDR	31:0	Flash Write Address. This field contains the target address for a write operation. A valid internal flash memory address is required for all write operations.	

12.2.2 Flash Clock Divide Register (FLSH_CLKDIV, Offset 0x0004)

Table 12-4. FLSH_CLKDIV (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	CLKDIV[7:0]							
Reset	0111 1000							
Access	RW							

Name	Bits	Description	Settings
CLKDIV	7:0	Flash Clock Divide. The clock is divided by this value to generate a 1 MHz clock for the Flash controller. The flash clock is only used during erase and write operations.	
RFU	31:8	Reserved.	N/A

12.2.3 Flash Control Register (FLSH_CN, Offset 0x0008)

Table 12-5. FLSH_CN (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	UNLOCK[3:0]				RFU	RFU[1:0]		PEND
Reset	0000				0	00		0
Access	RW				R	R		R
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	ERASE_CODE[7:0]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			RFU	RFU	PGE	ME	WR
Reset	000			0	0	0	0	0
Access	R			R	R	RS	RS	RS

Name	Bits	Description	Settings
WR	0	Write. This bit is automatically cleared after the operation.	0: No operation. 1: Initiate Write Operation.
ME	1	Mass Erase. This bit is automatically cleared after the operation.	0: No operation. 1: Initiate Mass Erase.
PGE	2	Page Erase. This bit is automatically cleared after the operation.	0: No operation. 1: Initiate Page Erase.
RFU	3	Reserved	N/A
RFU	4	Reserved	N/A
RFU	7:5	Reserved	N/A
ERASE_CODE	8:15	Erase Code. The ERASE_CODE must be set up properly before the erase operation can be initiated. These bits are automatically cleared after the operation is complete.	0x00 - No operation. 0x55 – Enable Page Erase. 0xAA – Enable Mass Erase. The debug port must be enabled.
RFU	23:16	Reserved	N/A
PEND	24	Flash Pending. When the Flash operation is in progress (busy), the Flash reads and writes fail. When PEND is set, write to all Flash registers, with exception of the Flash interrupt register, are ignored.	0: Flash Idle. 1: Flash Busy.
RFU	26:25	Reserved	N/A
RFU	27	Reserved	N/A
UNLOCK	31:28	Flash Unlock. The unlock code must be written to these four bits before any Flash write or erase operation is allowed.	0010: Flash unlocked. Others: Flash locked.

12.2.4 Flash Interrupt Register (FLSH_INT, Offset 0x0024)

Table 12-6. FLSH_INT (Offset 0x0024)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[5:0]						AFIE	DONEIE
Reset	0000 00						0	0
Access	R						RW	RW
Position	7	6	5	4	3	2	1	0
Field	RFU[5:0]						AF	DONE
Reset	0000 00						0	0
Access	R						RW0C	RW0C

Name	Bits	Description	Settings
DONE	0	Flash Done Interrupt. This bit is set to 1 upon Flash write or erase completion.	0: No Interrupt. 1: Interrupt pending. (Flash operation complete)
AF	1	Flash Access Failure Interrupt. This bit is set when an attempt is made to write the flash while the flash is busy, or the flash is locked. This bit can only be set to '1' by the hardware. This bit can be cleared by writing a '0'. Writing a '1' to this bit has no effect.	0: No failure. 1: Failure occurs.
RFU	7:2	Reserved.	N/A
DONEIE	8	Flash Done Interrupt Enable.	0: Disabled. 1: Enabled.
AFIE	9	Flash Access Fail Interrupt Enable.	0: Disabled. 1: Enabled.
RFU	31:10	Reserved.	N/A

12.2.5 Flash Data Register 0 (FLSH_DATA0, Offset 0x0030)

Table 12-7. FLSH_DATA0 (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	DATA0[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DATA0[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DATA0[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DATA0[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DATA0	31:0	Flash Write Data 0. Flash data for bits 31:0.	

12.2.6 Flash Data Register 1 (FLSH_DATA1, Offset 0x0034)

Table 12-8. FLSH_DATA1 (Offset 0x0034)

Position	31	30	29	28	27	26	25	24
Field	DATA1[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DATA1[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DATA1[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DATA1[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DATA1	31:0	Flash Write Data 1. Flash data for bits 63:32.	

12.2.7 Flash Data Register 2 (FLSH_DATA2, Offset 0x0038)

Table 12-9. FLSH_DATA2 (Offset 0x0038)

Position	31	30	29	28	27	26	25	24
Field	DATA2[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DATA2[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DATA2[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DATA2[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DATA2	31:0	Flash Write Data 2. Flash data for bits 95:64.	

12.2.8 Flash Data Register 3 (FLSH_DATA3, Offset 0x003C)

Table 12-10. FLSH_DATA3 (Offset 0x003C)

Position	31	30	29	28	27	26	25	24
Field	DATA3[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DATA3[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DATA3[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DATA3[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DATA3	31:0	Flash Write Data 3. Flash data for bits 127:96.	

12.2.1 Access Control Register (ACNTL, Offset 0x0040)

Table 12-11. ACNTL (Offset 0x0040)

Position	31	30	29	28	27	26	25	24
Field	ADATA[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	ADATA[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	ADATA[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	ADATA[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
ADATA	31:0	System Info Block Access Data	Writing the ACNTL register with the following values in the order shown, allows read and write access to the system and user information block: <pre>pflc->acntl = 0x3a7f5ca3; pflc->acntl = 0xa1e34f20; pflc->acntl = 0x9608b2c1;</pre> When unlocked, a write of any word disables access to the system and user information block. Readback of this register is always zero.

13.0 Inter-Integrated Circuit (I²C)

The I²C interface is a bidirectional, two-wire serial bus that provides a medium-speed communications network. It can operate as a one-to-one, one-to-many, or many-to-many communications medium.

The I²C supports both master and slave modes. This engine supports both standard-mode and fast-mode I²C standards. The slave engine shares the same I/O port as the master engine and is selected through the I/O configuration settings. It provides the following features:

- Master or slave-mode operation
- Supports standard (7-bit) addressing or 10-bit addressing
- Support for clock stretching to allow slower slave devices to operate on higher-speed busses
- Multiple transfer rates:
 - Standard mode: 100kbps
 - Fast mode: 400kbps
- Internal filter to reject noise spikes
- Transmitter and receiver FIFO depth of 16 bytes
- Supports DMA data transfer

13.1 Description

The I²C port is a half-duplex serial port that uses two lines (SDA for data, and SCL for clocks) for data transmission. Standard 100 kHz and fast 400 kHz transmit modes are supported. When a peripheral state changes from disable to enable, the peripheral internal state is restored to a known initial state. Regardless of the peripheral operating state, if a peripheral is disabled and then re-enabled, it should start at a known initial state. When an interrupt has an associated interrupt enable control, wakeup conditions are also qualified by the interrupt enable control. When a peripheral is disabled, none of its control have any effects on the device.

13.2 Protocol

The I²C protocol communication is a 2-wire serial transmission. It is a half-duplex protocol where data is transferred at various baud rates: up to 100 Kbits in standard mode, and up to 400 Kbits in fast mode. Each transfer is made of a start bit followed by eight data bits, an acknowledge bit sent by the receiver, and a stop bit. Data is sent with the MSB first.

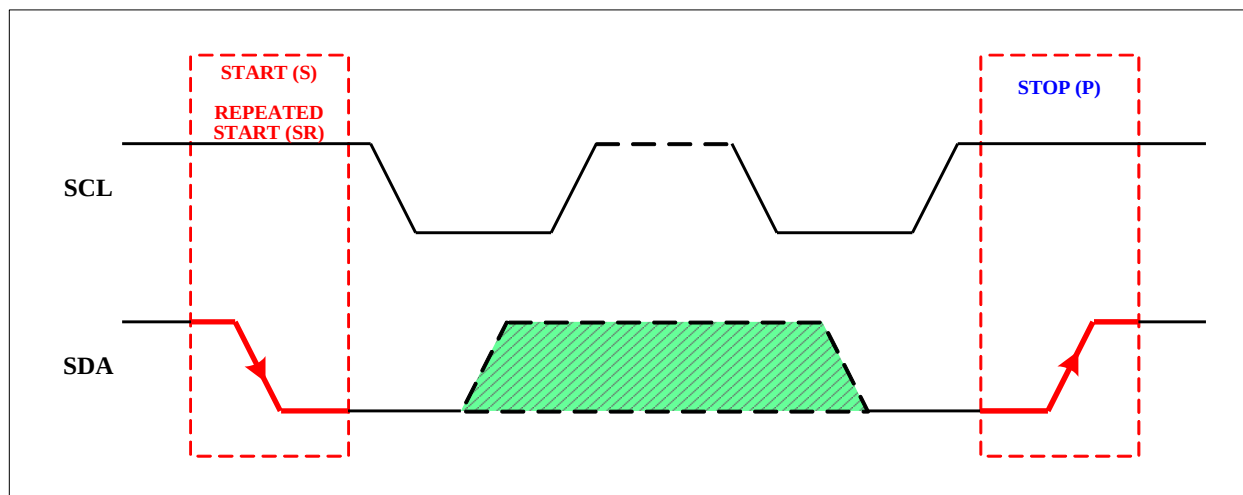
13.2.1 Bit Transfer

Both SDA and SCL lines are bi-directional lines connected to a positive supply voltage using a current source or a pullup resistor. When the bus is free, the lines are in high state. The data on the SDA line must be stable when the SCL line is high.

Communication starts when the SDA line switches from a high to low state, and the SCL line is high. Communication stops when the SDA line switches from a low to high state, and the SCL line is high. Only the master generates the start and stop conditions. During communication, the bus is considered busy.

The following figure shows the signals during communication.

Figure 12-1. I²C Start and Stop Conditions



13.2.2 Addressing

After a START or a repeated START condition, the address byte is transmitted as shown in Table 12-12.

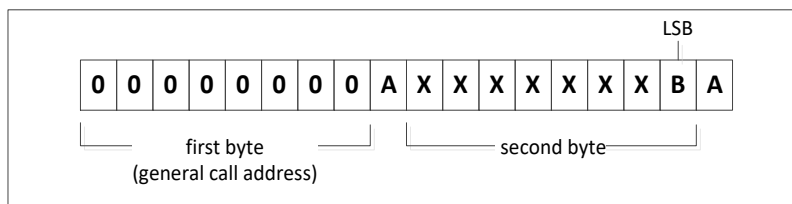
Table 12-12. Definition of the First Byte

7-1 bits	R/W bit	Description
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	Reserved (CBUS address)
0000 010	X	Reserved
0000 011	X	Reserved
0000 1XX	X	Reserved
1111 1XX	X	Reserved
1111 0XX	X	10- bit slave addressing

13.2.3 General Call Address

The General Call Address is 0x00. The acknowledge bit is optional if the device does not require the General Call Address data. A General Call Address is followed by another byte, which initiates all the devices. The format is given in General Call Address, Bytes Format. If General Call Enable (GCEN) is cleared to 0, then the general call address is ignored, and no acknowledgement is returned (leave SDA high). If GCEN=1, on receiving the general call address, the controller ACKs the general call address, and behaves as a slave-receiver. Both MST and CKMD are cleared to 0. General Call Interrupt is also set to 1 (GCI=1), and an interrupt is generated if enabled (GCIE=1).

Figure 12-2. General Call Address, Bytes Format



When bit B is 0, the second byte has the following definition:

0x06: reset and write the programmable part of the slave address by hardware.

0x04: write the programmable part of the slave address by hardware.

When bit B is set to 1, the hardware address of the master is written in the second byte.

The General Call Address has no effect on this interface.

13.2.4 START Byte

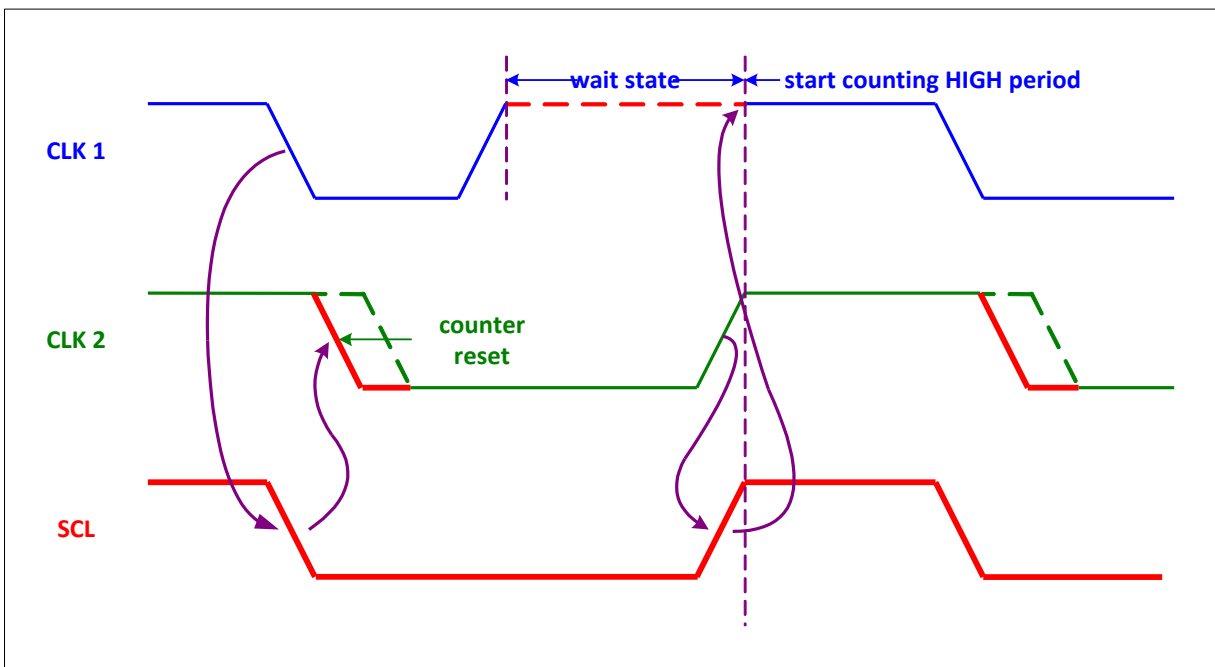
The START byte is used to initiate communication with slow devices. The START byte has no effect on this interface.

If two or more slaves are addressed at the same time, only the slave that has the smallest address (described in decimal figures) can exchange data with the master. Communication with the other slaves is cut off.

13.2.5 Clock Synchronization

The I²C protocol accepts multiple masters on the bus. This is why synchronization between the masters' clocks is compulsory. Clock synchronization is performed using the wired-AND connection of SCL. Figure 12-3 shows an example of clock synchronization.

Figure 12-3. Clock Synchronization



13.2.6 Clock Stretching

If a slave cannot receive or transmit another complete byte of data, it may hold the clock line SCL LOW to force the master into a wait state. Data transfer continues when the slave is ready for another byte of data and releases the SCL clock line from low.

The controller can hold SCL low in master and slave mode around the ACK bit. However, this is only clock stretching if done by a slave device. For a master, this is simply part of the generation of SCL, the speed of which can range from DC (0 Hz) up to the maximum f_{SCL} for a given bus speed. This section describes the situations in both the slave and master modes that lead to holding SCL low even though only the former is clock stretching as that term is defined in the I²C bus specification.

SCL is typically held low after the ACK bit and before the first data bit when the device has run out of either data to send or space to receive data depending on the transaction direction.

An exception is during interactive RX mode (IRXM), which performs clock stretching before the ACK bit on receive, and allows the software to decide whether to send an ACK or NACK. However, if operating in interactive receive

mode (IRXM=1) as a slave with clock stretching disabled (SCLSTRD=1), the SCL is not held low regardless of IRXMI. The SW must respond quickly enough to meet the data setup timing requirements. Setting ACK too late could even cause a transition on SDA while SCL is high, inadvertently generating a STOP or RESTART. So, do not use interactive receive mode with slave clock stretching disabled.

For a transmit operation as either slave or master, when the last byte is shifted out, and TX_FIFO is empty, SCL is held low until the TX_FIFO is written. Also, master transmitters can stop clock stretching on a TX_FIFO empty condition by setting STOP or RESTART, thereby ending the transaction. (A slave transmitter relies on the external master sending a NACK to indicate the end of the transaction. Otherwise, it does not know to release the SDA bus).

For a receive operation as either slave or master, when the RX_FIFO and shift register are full, SCL is held low until software reads from DATA (RX_FIFO). For a master receiver only, clock stretching also occurs once RXCNT bytes are received indicating the end of the transaction. However, the software has not yet instructed the controller how to proceed by setting either STOP or RESTART. So, where RXCNT is met, but there is no space in the RX_FIFO to store the final byte being held in the shift register, the controller waits for the software to read from the RX_FIFO before proceeding with any STOP or RESTART generation. Thus, the software must read at least one byte from the RX_FIFO at which point the controller can then save off the last byte and generate STOP or RESTART if indicated to do so. Note: A slave receiver does not stop receiving until the external master sends a STOP or RESTART condition during what would be the first data bit. Because the master has control of SDA during data transmission, there is no need for NACK to instruct the slave to release SDA.

Clock stretching in slave mode may be completely disabled by setting the I2CCN.SCLSTRD bit. This feature is for systems where the external I²C masters do not support slave clock stretching. The generation of SCL in master mode transaction is unaffected by this bit; the controller continues to hold SCL low in the above conditions.

For a slave transmit operation with clock stretching disabled, when the last byte is shifted out, and TX_FIFO is empty, 0xFF is shifted out for all remaining bytes until the end of the transaction. To avoid this situation, you can preload the TX_FIFO prior to the Slave Address Match. See the section on FIFO Management for more information. Note that the I²C bus specification does not provide any mechanism in this case for the slave to indicate this error condition to the master because a transmitter does not send ACK bits (it only receives them). The onus is on the software, application, and system designers to handle this condition appropriately. Possible options include:

- Ensuring this condition never happens by preloading the TX_FIFO with data, using DMA, or both.
- Signaling the data loss occurrence to the master through some higher-layer protocol or messaging system.

For a slave receive operation with clock stretching disabled, when the RX_FIFO and the shift register are full, the last incoming data byte is lost. The controller sends a NACK to the master to indicate this data-loss condition.

13.2.7 Timeout

The Timeout function is provided to indicate a transfer-bus error. The Timeout register determines the timeout value using the following equation:

$$\text{Timeout} = \text{I2C\_CLKIN} * (\text{TO} * 8 + 3)$$

The timeout function is enabled when TO is not zero. When enabled, the Timeout timer starts counting when SCL is driven low by hardware, and resets when SCL is released. The timeout counter only monitors if the chip is driving SCL low, not if an external device is holding it low. It does not monitor the status of SDA.

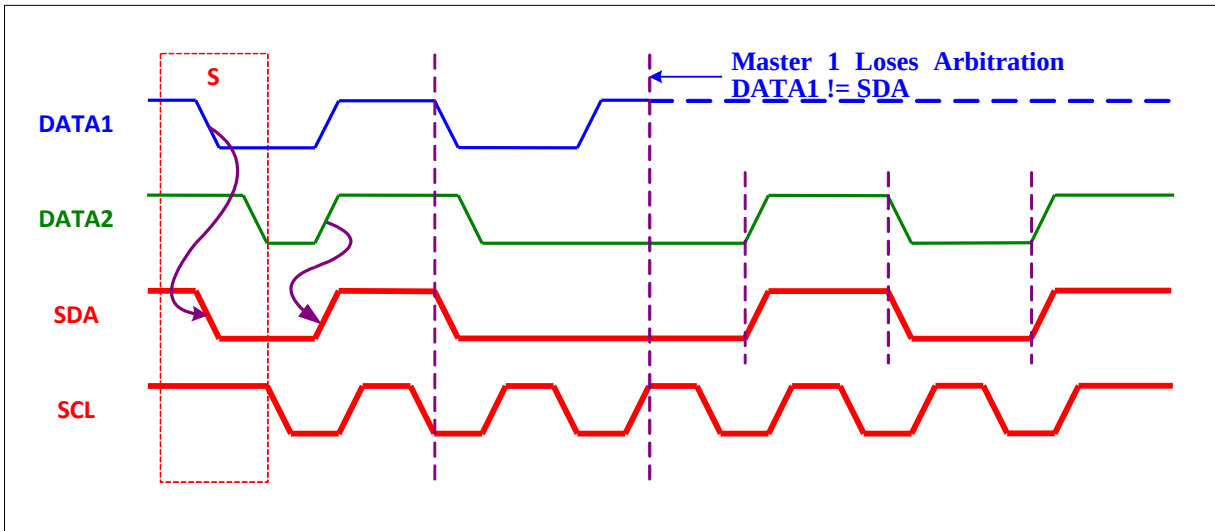
On timer expiration, the controller concludes that there is a bus error, and sets ERRI to 1. The I²C Bus Status (BUS) continues to reflect the current bus status.

For applications where the slave might hold the SCL line longer than the timeout value allowed, you can disable functions by clearing TO to 0.

13.2.8 Bus Arbitration

When the bus is free, two or more masters can initiate communication, but only one master can make a valid transmission, and the others must switch to slave mode. Each master compares the data sent to the SDA line. If the data is different, the master that output the HIGH state on the SDA line switches off its SDA data output. Arbitration occurs until only one master remains. Figure 12-4 shows a master arbitration.

Figure 12-4. Master Arbitration



13.3 FIFO

The transmit FIFO (TX_FIFO) and receive FIFO (RX_FIFO) are separate. However, both are accessed using the DATA register. Reading from DATA returns data from RX_FIFO. Writing to DATA writes to TX_FIFO. FIFOs are shared between the master and slave. Both the RX_FIFO and TX_FIFO are flushed when I2CEN=0.

When operating as a receiver, Receive Threshold (RXTH) defines the number of bytes that sets the Receive Threshold Interrupt (RXTHI). When the number of data bytes in the RX_FIFO is RXTH, RXTHI is set, and an interrupt is generated if enabled (RXTHIE=1).

Flush the RX_FIFO by setting Receive FIFO Flush (RXFSH) to 1. Reads from an empty RX_FIFO return 0xFF, but otherwise have no effect.

When operating as a transmitter, Transmit Threshold (TXTH) defines the number of bytes that sets the Transmit Threshold Interrupt. When the number of data bytes in the TX_FIFO is TXTH, TXTHI is set, and an interrupt is generated if enabled (TXTHIE=1).

Writes to a full TX_FIFO are ignored.

Flush the TX_FIFO by setting Transmit FIFO Flush (TXFSH) to 1. Also, the TX_FIFO is also flushed in the following conditions:

- General Call Address match (GCI), and TXPRELD is not set
- Slave Address match (AMI), and TXPRELD is not set
- Operating as a slave transmitter, and a NACK is received
- Any of the following interrupts: ARBERI, TOERI, ADRERI, DATERI, STRTERI, or STOPERI

When the above conditions occur, the Transmit Lockout Interrupt (TXLOI) is set to 1. When TXLOI is set, the TX_FIFO is flushed, and any writes to the TX_FIFO are ignored. The TXLOI might be simultaneously or independently cleared with the interrupt signals. The TXLOI gets set again only after all the sources are cleared, and any one of the conditions occurs again.

The flushing of the TX FIFO on Slave Address Match or General Call Match might be disabled using the Transmit FIFO Preload bit (I2CTXCFG.TXPRLD). Setting this bit allows high-speed applications to preload the Transmit FIFO prior to Slave Address Match.

13.4 Interrupt

The I²C Controller contains multiple interrupt sources that are combined into one interrupt request signal to the interrupt controller. If the I²C Controller is enabled, the source of the interrupt is determined by which bits are set in the I2CINT0 register.

It is recommended to acknowledge an interrupt before enabling it (unmask operation) to avoid a previous event triggering.

13.5 Interactive Receive Mode

In some communication protocols, the device might want to inspect portions of the received data, and determine the corresponding response. In this case, Interactive Receive Mode can be used. This mode is enabled by setting the Interactive Receive Mode (IRXM) bit to 1. The Interactive Receive Mode must be set to 1 before any I²C transfer begins.

When IRXM=1, the device automatically holds the clock low before the ACK bit after every data byte. It also sets the Interactive Receive Mode Interrupt (IRXMI=1). This can generate an interrupt if enabled (IRMXIE=1). The IRXMI flag is set to 1 after the data byte 8th SCL falling edge. The software can read the data and determine the appropriate ACK value by writing to the ACK bit. If ACK=1, the device acknowledges with a NACK (leaving SDA high). If ACK=0, the device acknowledges with an ACK (pulling SDA low). After setting the ACK value, writing 1 to clear IRXMI allows you to output the ACK bit value onto SDA. For both master and slave operations, SCL is released only after the necessary SCL low time requirement is satisfied, so as to avoid violating timing.

A NACK from the receiver indicates to the transmitter that the receiver has received all the bytes it expects from the transmitter, or an error has occurred, and the receiver no longer accepts any more data. The slave should release SDA line on the following SCL low and allow the master to drive to generate a STOP or repeated START condition.

While the controller is waiting for IRXMI to be cleared, software can clear IRXM, and if operating as a master, load a value into RXCNT. This allows the software to examine the initial bytes of a transaction manually (which might be a command), and then switch to automatic reception for the remaining bytes. Note that when switching out of IRXM, the software must read the last data byte before IRXM is cleared to 0. Otherwise, the software simply reads the 0xFF output from the empty RX_FIFO.

The Interactive Receive Mode does not apply to address bytes; only to data bytes.

The Interactive Receive Mode does not apply to general call address responses or START byte responses.

13.6 Error

When an error occurs during a transaction, the corresponding error interrupt bit is set to 1, and the TX_FIFO is flushed. Writes to the TX_FIFO are also ignored (TXLOI=1).

Depending on the error, the transaction is also terminated. To respond to the error condition, software can control SCL and SDA directly by setting Software Output Enable (SWOEN) to 1. When enabled, SCL and SDA output are controlled by SCLO and SDAO, respectively. Note that the internal controller is also affected by the state of SCL and SDA and might be in an undefined state. Therefore, after manually controlling SCL/SDA, disable the controller (I2CEN=0), and reenable (I2CEN=1) to reset the controller to a known initial state.

13.7 DMA

The I²C supports the DMA peripheral communication with the Lighthouse family of DMA controllers. It allows the DMA to read, write, or both the FIFOs of the I²C device.

DMA activity is triggered by the transmit (I2CTXCFG.TXTH) and receive (I2CRXCFG.RXTH) FIFO thresholds.

When the TX FIFO byte count (I2CTX.TXFIFO) is less than or equal to I2CTXCFG.TXTH, the DMA transfers data to the TX FIFO according to the DMA configuration.

When the RX FIFO count (I2CRX.RXFIFO) is greater than or equal to I2CRXCFG.RXTH, the DMA fetches data from the RX FIFO and transfers it according to the DMA configuration.

To enable the DMA transfers, enable the channel enables for transmit (I2CDMA.TXEN) and/or receive (I2CDMA.RXEN), and refer to the DMA chapter in this document. If you choose to enable the DMA, there is no special configuration for the I²C device: the DMA is a core (executing software) replacement. The I²C interrupts are not rerouted to the DMA and are still routed to the core (it is up to the software to configure the interrupts according to the DMA usage).

13.8 Master Operation

The peripheral operates in master mode when the Master Mode (MST) is set to 1. The master operation is only valid when the device is operating in the master mode (MST=1).

To initiate a transfer, the master generates a START (S) condition by setting the START bit to 1 (START=1). If the bus is busy, the master waits until the bus is available to generate a START.

A master can communicate with two slave devices without relinquishing the bus. Instead of generating a STOP condition after communicating with the first slave, the master generates a repeated START (SR) by setting the Repeated START bit (RESTART) to 1. If a transaction is in progress, the master finishes the current transaction before generating the repeated START. The controller then starts transmitting the slave address from the next bytes in the TX_FIFO. The RESTART bit is cleared to 0 as soon as the master starts RESTART generation. The reception of a STOP condition also clears any pending RESTART.

The START bit is automatically cleared to 0 after the master has completed the transaction and successfully generated the STOP condition.

The master can also generate a STOP condition by setting the STOP bit (STOP) to 1.

If both START and RESTART are set at the same time, the START condition is generated first. Then, at the end of the first transaction, a RESTART condition is generated. For a transmit transaction, the first transaction ends when the TX_FIFO is empty, or a slave responds with a NACK. For a receive transaction, the first transaction ends when the master has received RXCNT bytes.

If both RESTART and STOP are set at the same time, a RESTART condition is generated, and both bits are cleared (that is, the STOP bit is effectively ignored).

If START, RESTART and STOP are set at the same time, a START condition is generated first. Then, at the end of the first transaction, a RESTART condition is generated. The STOP bit is cleared and effectively ignored.

To operate as Hs-mode master, set Hs-mode Enable (HSM DEN) to 1. Instead of shifting data out from TX_FIFO after generating the START condition, the master shifts out its Hs-Mode Master Code (MCODE). After winning arbitration, it switches to Hs-mode and sends a repeated START condition and then a slave address from TX_FIFO. The software does not need to write the HS prefix and master code to TX_FIFO; the controller sends this out automatically, generates the necessary RESTART, and then transmits the slave address stored in the TX_FIFO.

The slave cannot generate START, repeated START, or STOP conditions. Therefore, when MST=0, START, RESTART and STOP are all cleared to 0.

Once a transfer has started (START=1), do not change the operation mode or setting or unpredictable behavior might occur.

13.8.1 Arbitration

The I²C bus is free after a STOP condition is generated. Any master on the bus can initiate a transfer when the bus is idle. It is possible that two or more masters might start transmitting during the minimum hold time of the START condition (t_{HD_STA}). This is a valid situation.

If two or more masters try to put information onto the bus, the first to produce a 1 on SDA when the other produces a zero loses arbitration.

In this implementation, the master stops generating the clock signal once it has lost arbitration. It also clears the START, RESTART, STOP to 0, set ARBERI to 1, and set TXLOI to 1. Because the I²C module can operate in slave mode as well, the losing master continues to shift in address bytes and compare against its own address (SLA). If matched, it sets the address match flag to 1 (AMI=1) and generates an interrupt if enabled. If not addressed, the controller ignores the rest of the transaction until the next START/RESTART. Note that arbitration loss alone does not clear the MST bit: there must be an address match (AMI or GCI) to clear MST. Any address match clears the MST bit regardless of whether or not it was preceded by an arbitration loss.

When operating as a slave, the controller aborts the transaction and sets ARBERI to 1 when detecting arbitration loss.

13.8.2 Clock Generation

The master is responsible for generating the clock on bus SCL. The I²C clock base is supplied by I2C_CLKIN.

The SCL high time is defined by I²C Clock High Register (I²CCKH) whereas the SCL low time is defined by I²C Clock Low Register (I2C_CKL).

$$\text{SCL High} = \text{I2C\_CLKIN} * (\text{CKH}+1)$$

$$\text{SCL Low} = \text{I2C\_CLKIN} * (\text{CKL}+1)$$

During Synchronization, when external masters or external slaves are driving SCL simultaneously, the actual SCL duty cycle is affected. By monitoring SCL state, the controller can determine whether an external master or slave is holding SCL low. In all cases, the controller waits until the SCL has gone high before starting to count the SCL high cycles. Similarly, if an external master pulls SCL low before the controller has finished counting its I2CCKH cycles, the controller starts counting its I2CKL cycles and releases SCL once the I2CCKL count has expired.

Note that the actual clock timing can be longer due to bus loading, other devices holding the clock, or the input glitch filter delay time.

13.8.3 Master-Transmitter

A master transmit operation consists of the following sequence:

1. Write the Target Slave Address bytes to DATA (TX_FIFO) with the R/W bit cleared to 0.
2. Write the data bytes to DATA(TX_FIFO).
3. Set the START bit.
4. Wait for the DONEI or the TXTHI interrupt.

The controller waits for the bus to become idle and then generates a START condition.

On successful START generation, the controller sends out the slave address from TX_FIFO and continues to shift data out until TX_FIFO is empty.

13.8.3.1 Preloading the TX FIFO for transmission

There are situations where the slave wants to preload the TX_FIFO prior to transmission. An address match or a general call match clears the TX_FIFO. This is not always the best option particularly if the clock stretching is disabled. The software can also want to support acknowledge polling from an external master, and send a NACK to a

read request from the master until the requested data is ready. For either of these cases, software should enable TX FIFO Preloading.

To allow preloading of the TX FIFO, set I2CTXCFG.TXPREDL = 1. This value suppresses flushing of the TX_FIFO under these conditions. Error conditions and receiving a NACK as a slave transmitter still cause the setting of TXLOI and still flush the TX_FIFO.

For software to signal hardware that the controller is ready for a slave transmit operation, the TXRDY bit is available. Once the software sets TXPRELD=1, TXRDY is automatically cleared to 0. Until software sets TXRDY back to 1, the controller NACKs any address match for a transmit operation (i.e., R/W = 1). Once the software has loaded the transmit FIFO, write this bit to 1 to signal that the TX_FIFO contents are ready for transmission. The controller then ACKs the next transmit operation from the master. After the transmission has started, hardware automatically clears this bit once the first byte is shifted out. TXRDY is also automatically cleared whenever the TX_FIFO is flushed. Writing a 0 to TXRDY has no effect. When TX FIFO preloading is disabled (TXPRELD=0), TXRDY is automatically forced to 1 by the hardware, and GCI and AMI flush the TX_FIFO.

13.8.4 Master-Receiver

A master receiver operation consists of the following sequence:

1. Write number of received data bytes to RXCNT.
2. Write Target Slave Address bytes to DATA (TX_FIFO) with the R/W bit set to 1.
3. Set the START bit.
4. Wait for DONEI or RXTHI Interrupt.

The controller waits for the bus to be idle and then generates a START condition.

On successful START generation, the controller sends out the slave address from TX_FIFO. On receiving the ACK from the slave, the controller starts shifting the SDA data in. For proper operation, set RXCNT prior to transaction and keep it constant until the DONEI flag is set.

If Interactive Receive Mode is disabled (IRXM=0), and STOP=RESTART=0, the controller holds SCL low after RXCNT. By default, the controller returns ACK after every byte except the last (RXCNT). On the last byte, the controller returns NACK and sets DONEI to 1. If STOP=1, the controller also generates a STOP after RXCNT data bytes.

If SCL is held low due to RXCNT fulfilled but both STOP=RESTART=0, software generates a repeated START or a STOP condition. If repeated START is generated, set RESTART to 1. Note that after repeated STARTs are generated, the controller waits for data available in TX_FIFO before outputting any SCL. The software should update RXCNT before writing to TX_FIFO for another master-receiver operation. Set STOP=1 to generate a STOP.

If interactive received mode is enabled (IRXM=1), then software should also set the RESTART/ STOP bit prior to setting the NACK response. If not, after the NACK is shifted out, the controller continues to hold SCL low until either RESTART or STOP is set to 1.

13.9 Slave Operation

The peripheral operates in slave mode when the Master Mode Bit (MST) bit is cleared to 0 or when addressed as a slave (AMI=1 or GCEN=GCI=1). Because the slave does not drive SCL clock cycles, Clock Mode (CKMD) is cleared to 0.

13.9.1 Slave-Transmitter

The device operates as a slave transmitter when an address matched with R/W bit is set to 1. The external master is reading from the slave.

If TX_FIFO is not empty, then the TX_FIFO contents are clocked out by SCL. The master sends a NACK to the slave when it requires no more data from the slave. On receiving a NACK from the master, TXLOI is set, automatically flushing the TX_FIFO and setting DONEI to 1.

Normally, when the last byte is shifted out, and TX_FIFO is empty, the slave holds SCL low until TX_FIFO is written. However, if the TXLAST bit is set to 1, SCL is released after the last byte is shifted out, and TX_FIFO is empty. TXLAST is automatically cleared to 0 when TXLOI=1.

Once a slave starts transmitting out of its TX_FIFO, detection of out of sequence STOP, START, or RESTART conditions terminates the current transaction. When a transaction is terminated in this manner, either STRTERI or STOPERI is set to 1.

13.9.2 Slave-Receiver

The device operates as a slave receiver when an address matched with the R/W bit is cleared to 0. The external master is writing to the slave. When an address match occurs, the data is clocked into RX_FIFO, and an ACK is returned to the master unless instructed otherwise.

A transaction finishes when the device detects a STOP or RESTART condition. It also sets DONEI to 1.

When IRXMI=1, on every data byte, SCL is held low prior to an ACK and waits for an ACK and IRXMI to clear. If a NACK is returned, then the transaction finishes, and DONEI is set to 1. Once the slave returns NACK, any following bytes are ignored and NACKed.

Once a slave starts receiving data into its RX_FIFO, the detection of out of sequence STOP, START, or RESTART conditions releases the I²C bus to the idle state and sets ERRI to 1. Note that RX_FIFO is not flushed.

13.10 I²C Registers

Address assignments for I²C registers within each block are outlined in Table 12-13. Write reserved register bits only as 0.

Table 12-13. I²C Register Address (Base ADDR = 0x4001_D000)

Offset	Access	Register	Description
0x0000	RW	I2CCN	Control Register
0x0004	RW	I2CST	Status Register
0x0008	RW	I2CINT0	Interrupt Status 0 Register
0x000C	RW	I2CINTEN0	Interrupt Enable 0 Register
0x0010	RW	I2CINT1	Interrupt Status 1 Register
0x0014	RW	I2CINTEN1	Interrupt Enable 1 Register
0x0018	R	I2CFIFO	FIFO Configuration Register
0x001C	RW	I2CRXCFG	Receive Configuration Register
0x0020	RW	I2CRX	Receive Register
0x0024	RW	I2CTXCFG	Transmit Configuration Register
0x0028	RW	I2CTX	Transmit Register
0x002C	RW	I2CDATA	DATA Register
0x0030	RW	I2CMCN	Master control Register
0x0034	RW	I2CCKL	Clock Low Register
0x0038	RW	I2CCKH	Clock High Register
0x003C	R	Reserved	Reserved
0x0040	RW	I2CTO	Timeout Register
0x0044	RW	I2CSLA	Slave Address Register
0x0048	RW	I2CDMA	DMA Register

Table 12-14. Function Control Register Address (Base ADDR = 0x4000_0800)

Offset	Access	Register	Description
0x0000	RW	I2CDGLTCHR	I2C Deglitcher Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details about the reset values and various access modes.

13.10.1 I²C Control Register (I2CCN, Offset 0x0000)

Table 12-15. I2CCN (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field				RFU[7:0]				
Reset				0000 0000				
Access				R				
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU	RFU	SCLPPM	SCLSTRD	READ	SWOE	SDA	SCL
Reset	0	0	0	0	0	0	-	-
Access	R	R	RW	RW	R	RW	R	R
Position	7	6	5	4	3	2	1	0
Field	SDAO	SCLO	RFU	ACK	IRXM	GCEN	MST	I2CEN
Reset	0	0	0	0	0	0	0	0
Access	RW	RW	R	RW	RW	RW	RW	RW

Name	Bits	Description	Settings
I2CEN	0	I ² C Enable.	0: I ² C Disabled 1: I ² C Enabled
MST	1	Master Mode Enable	0: Slave Mode 1: Master Mode
GCEN	2	General Call Enable.	0: Ignore General Call address 1: Acknowledge General Call address
IRXM	3	Interactive Receive Mode	0: Disabled 1: Enabled
ACK	4	Data Acknowledge. This bit defines the acknowledge bit returned by the I ² C receiver while IRXM=1. Hardware forces ACK to 0 when IRXM = 0.	0: Return ACK (pulling SDA Low) 1: Return NACK (leaving SDA High)
RFU	5	Reserved	N/A
SCLO	6	SCL Output. This bit controls SCL output when SWOE=1	0: Drive SCL Low 1: Release SCL
SDAO	7	SDA Output. This bit controls SDA output when SWOE=1	0: Drive SDA Low 1: Release SDA
SCL	8	SCL Status. This bit reflects the logic state of the SCL signal. This bit is read only.	
SDA	9	SDA Status. This bit reflects the logic state of the SDA signal. This bit is read only.	
SWOE	10	Software output Enabled.	0: I ² C Outputs SCLO and SDAO Disabled 1: I ² C Outputs SCLO and SDAO Enabled
READ	11	Read. This bit reflects the R/W bit on an address match (AMI=1) or general call match (GCI=1). This bit is valid three cycles after the relevant interrupt bit is set.	0: Write 1: Read
SCLSTRD	12	SCL Stretch Disable. This bit disables slave-clock stretching when set.	0: Slave clock stretching enabled 1: Slave clock stretching disabled

Name	Bits	Description	Settings
SCLPPM	13	SCL Push-Pull Mode. This bit controls whether SCL is operated in the I ² C standard open-drain mode, or in a non-standard push-pull mode where the Hi-Z output is replaced with Drive-1. Only use the non-standard mode when operating as a master and communicating with slaves guaranteed to never drive SCL low.	0: Standard open-drain operation: drive low for 0, Hi-Z for 1 1: Non-standard push-pull operation: drive low for 0, drive high for 1
RFU	31:14	Reserved	N/A

13.10.2 I²C Status Register (I2CST, Offset 0x0004)

Table 12-16. I2CST (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				ST[3:0]			
Reset	0000				0000			
Access	R				R			
Position	7	6	5	4	3	2	1	0
Field	RFU	RFU	CKMD	TXF	TXE	RXF	RXE	BUS
Reset	0	0	0	0	1	0	1	0
Access	R	R	R	R	R	R	R	R

Name	Bits	Description	Settings
BUS	0	Busy. The I ² C bus is busy when another master is using the bus, or this interface itself is using the bus.	0: The bus is idle. 1: The bus is busy.
RXE	1	RX Empty. This bit is read only and set to 1 on a RX_FIFO flush and when I2CEN = 0.	0: Not Empty 1: Empty
RXF	2	RX Full. This bit is read only and cleared to 0 on a RX_FIFO flush and when I2CEN = 0.	0: Not Full 1: Full
TXE	3	TX Empty. This bit is read only and set to 1 on a TX_FIFO flush and when I2CEN = 0.	0: Not Empty 1: Empty
TXF	4	TX Full. This bit is read only and cleared to 0 on a TX_FIFO flush and when I2CEN = 0.	0: Not Full 1: Full
CKMD	5	Clock Mode.	0: Device not actively driving SCL clock cycles. 1: Device operating as master and actively driving SCL clock cycles.
RFU	7:6	Reserved	N/A

Name	Bits	Description	Settings
ST	11:8	Status Controller	0000: Controller Idle 0001: Master Transmit address 0010: Master Receive address ACK 0011: Master Transmit extended address 0100: Master Receive extended address ACK 0101: Slave Receive address 0110: Slave Transmit address ACK 0111: Slave Receive extended address 1000: Slave Transmit extended address ACK 1001: Transmit data (master or slave) 1010: Receive data ACK (master or slave) 1011: Receive data (master or slave) 1100: Transmit data ACK 1101: NACK stage (master or slave) 1110: Reserved 1111: Bystander state (ongoing transaction but not participant-another master addressing another slave.
RFU	31:12	Reserved	N/A

13.10.3 I²C Interrupt 0 Register (I2CINT0, Offset 0x0008)

Table 12-17. I2CINT0 (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	TXLOI	STOPERI	STRTERI	DNRERI	DATERI	ADRERI	TOERI	ARBERI
Reset	0	0	0	0	0	0	0	0
Access	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
Position	7	6	5	4	3	2	1	0
Field	ADRACKI	STOPI	TXTHI	RXTHI	AMI	GCI	IRXMI	DONEI
Reset	0	0	1	1	0	0	0	0
Access	RW1C	RW1C	R	R	RW1C	RW1C	RW1C	RW1C

Name	Bits	Description	Settings
DONEI	0	Transfer Done Interrupt	0: Interrupt not pending 1: Transfer done
IRXMI	1	Interactive Receive Interrupt. This flag is set by HW only after the 8th bit when IRXM=1.	0: ACK valid 1: ACK invalid
GCI	2	Slave General Call Address Match Interrupt	0: Interrupt not pending 1: General call address received when GCEN=1
AMI	3	Slave Address Match Interrupt	0: Interrupt not pending 1: Address matched detected
RXTHI	4	Receive Threshold Interrupt. This bit is automatically cleared when RX_FIFO is below the threshold level.	0: Interrupt not pending 1: Interrupt pending. RX_FIFO equal or more bytes than the threshold.
TXTHI	5	Transmit Threshold Interrupt. This bit is automatically cleared when TX_FIFO is above the threshold level.	0: Interrupt not pending 1: Interrupt pending. TX_FIFO has equal or less bytes than the threshold.
STOPI	6	STOP Interrupt	0: Interrupt not pending 1: STOP conditions detected when the device participates in a transaction
ADRACKI	7	Address Acknowledge Interrupt	0: Interrupt not pending 1: Received ACK from slave when sending address.
ARBERI	8	Arbitration Error Interrupt	0: Interrupt not pending 1: Arbitration loss detected in master mode.
TOERI	9	Timeout Error Interrupt	0: Interrupt not pending 1: Timeout detected. Bus stuck (SCL is held low by this device in master or slave mode)
ADRERI	10	Address NACK Error Interrupt	0: Interrupt not pending 1: Address NACK received in master mode.
DATERI	11	Data NACK Error Interrupt	0: Interrupt not pending 1: Data NACK received in master mode.

Name	Bits	Description	Settings
DNRERI	12	Do Not Respond Error Interrupt	0: Interrupt not pending 1: Receive address match when RX_FIFO or TX_FIFO is not ready in slave mode.
STRTERI	13	Start Error Interrupt. When in master mode, this error condition also clears I2CCN.MST and the STOP/START/RESTART flags.	0: Interrupt not pending 1: Out of Sequence Start / repeated Start Detected (master or slave mode).
STOPERI	14	Stop Error Interrupt. When in master mode, this error condition also clears I2CCN.MST and the STOP/START/RESTART flags.	0: Interrupt not pending 1: Out of Sequence Stop detected (master or slave mode).
TXLOI	15	Transmit Lock Out Interrupt	0: Allow write to TX_FIFO 1: Flush TX_FIFO, and ignore writes to TX_FIFO
RFU	31:16	Reserved	N/A

13.10.4 I²C Interrupt Enable 0 Register (I2CINTEN0, Offset 0x000C)

Table 12-18. I2CINTEN0 (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	TXLOIE	STOPERIE	STRTERIE	DNRERIE	DATERIE	ADRERIE	TOERIE	ARBERIE
Reset	0	0	0	0	0	0	0	0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Position	7	6	5	4	3	2	1	0
Field	ADRACKIE	STOPIE	TXTHIE	RXTHIE	AMIE	GCIE	IRXMIE	DONEIE
Reset	0	0	0	0	0	0	0	0
Access	RW	RW	RW	RW	RW	RW	RW	RW

Name	Bits	Description	Settings
DONEIE	0	Transfer Done Interrupt Enable.	0: Interrupt Disabled 1: Interrupt Enabled when DONEI = 1
IRXMIE	1	Interactive Receive Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled when IRXMI = 1
GCIE	2	Slave General Call Address Match Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled when GCI = 1
AMIE	3	Slave Address Match Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled when AMI = 1
RXTHIE	4	Receive Threshold Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
TXTHIE	5	Transmit Threshold Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
STOPIE	6	STOP Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled when STOPI = 1
ADRACKIE	7	Address Acknowledge Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
ARBERIE	8	Arbitration Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
TOERIE	9	Timeout Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
ADRERIE	10	Address NACK Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
DATERIE	11	Data NACK Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
DNRERIE	12	Do Not Respond Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
STRTERIE	13	Start Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled

Name	Bits	Description	Settings
STOPERIE	14	Stop Error Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled
TXLOIE	15	Transmit Lock Out Interrupt Enabled	0: Interrupt Disabled 1: Interrupt Enabled when TXLO = 1
RFU	31:16	Reserved	N/A

13.10.5 I²C Interrupt 1 Register (I2CINT1, Offset 0x0010)

Table 12-19. I2CINT1 (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[5:0]						TXUFI	RXOFI
Reset	0000 00						0	0
Access	R						RW1C	RW1C

Name	Bits	Description	Settings
RXOFI	0	Receive Overflow Interrupt. When operating as a slave receiver, this bit is set when you reach the first data bit, the RX FIFO bit, and the shift registers are both full.	0: No interrupt is pending 1: Event occurred.
TXUFI	1	Transmit Underflow Interrupt. When operating as a slave transmitter, this bit is set when you reach the first data bit, the TX FIFO is empty, and the master is still asking for more data (i.e., the master has not sent a NACK yet).	0: No interrupt is pending 1: Event occurred.
RFU	31:2	Reserved	N/A

Note: These bits are set by the hardware. Clear these bits by the software to acknowledge the interrupts.

13.10.6 I²C Interrupt Enable 1 Register (I2CINTEN1, Offset 0x0014)

Table 12-20. I2CINTEN1 (Offset 0x0014)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[5:0]						TXUFIE	RXOFIE
Reset	0000 00						0	0
Access	R						RW1C	RW1C

Name	Bits	Description	Settings
RXOFIE	0	Receive Overflow Interrupt Enable.	0: Interrupt disabled 1: Interrupt enabled
TXUFIE	1	Transmit Underflow Interrupt Enable.	0: Interrupt disabled 1: Interrupt enabled
RFU	31:2	Reserved	N/A

13.10.7 I²C FIFO Configuration Register (I2CFIFO, Offset 0x0018)

Table 12-21. I2CFIFO (Offset 0x0018)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	TXLEN[15:8]							
Reset	0000 1000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RXLEN[7:0]							
Reset	0000 1000							
Access	R							

Name	Bits	Description	Settings
RXLEN	7:0	Receive FIFO Length	Reads back 0x08
TXLEN	15:8	Transmit FIFO Length	Reads back 0x08
RFU	31:16	Reserved	N/A

13.10.8 I²C Receive Configuration Register (I2CRXCFG, Offset 0x001C)

Table 12-22. I2CRXCFG (Offset 0x001C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				RXTH[3:0]			
Reset	0000				0000			
Access	R				RW			
Position	7	6	5	4	3	2	1	0
Field	RXFSH	RFU	RFU	RFU	RFU	RFU	RFU	DNR
Reset	0	0	0	0	0	0	0	0
Access	W	R	R	R	R	R	R	RW

Name	Bits	Description	Settings
DNR	0	Do Not Respond	0: Always respond to address match 1: Do not respond to address match when RX_FIFO is not empty.
RFU	6:1	Reserved	N/A
RXFSH	7	Receive FIFO flush	0: FIFO not flushed. 1: Flush RX_FIFO. This bit is automatically cleared to 0 after the operation. Setting this bit to 1 affects the RX_FIFO status.
RXTH	11:8	Receive FIFO Threshold. These bits define the RX_FIFO interrupt threshold. Minimum FIFO length is 1, and maximum is 7.	
RFU	31:12	Reserved	N/A

13.10.9 I²C Receive Register (I2CRX, Offset 0x0020)

Table 12-23. I2CRX (Offset 0x0020)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				RXFIFO3:0]			
Reset	0000				0000			
Access	R				R			
Position	7	6	5	4	3	2	1	0
Field	RXCNT[7:0]							
Reset	0000 0001							
Access	RW							

Name	Bits	Description	Settings
RXCNT	7:0	Receive Count Bits. These bits define the number of bytes received in a transaction except when RXCNT=0. RXCNT=0 means 256 bytes are received in a transaction.	
RXFIFO	11:8	Receive FIFO Count. These bits reflect the number of bytes in the RX_FIFO. These bits are read only and cleared to 0 when the RX_FIFO is flushed, and when I2CEN=0.	
RFU	31:12	Reserved	N/A

13.10.10 I²C Transmit Configuration Register (I2CTXCFG, Offset 0x0024)

Table 12-24. I2CTXCFG (Offset 0x0024)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				TXTH[3:0]			
Reset	0000				0000			
Access	R				RW			
Position	7	6	5	4	3	2	1	0
Field	TXFSH	RFU	RFU	RFU	RFU	RFU	TXRDYMODE	TXPRELD
Reset	0	0	0	0	0	0	0	0
Access	W	R	R	R	R	R	RW	RW

Name	Bits	Description	Settings
TXPRELD	0	Transmit FIFO Preload Mode. When TXPRELD is set, address match or general call match does not set TX Lockout nor clear the TX FIFO. This allows the software to preload data into the FIFO, setup DMA in preparation for subsequent transactions, or both.	
TXRDYMODE	1	Transmit FIFO Ready Manual Mode	0: Hardware control of I2CTXRDY enabled. 1: Hardware control of I2CTXRDY disabled. This allows software to handle this function if necessary.
RFU	6:2	Reserved	N/A
TXFSH	7	Transmit FIFO flush. This bit also reads back 1 when the TX FIFO is being flushed due to TXLOI=1.	0: FIFO not flushed. 1: Flush TX_FIFO. This bit is automatically cleared to 0 after the operation. Setting this bit to 1 affects the TX_FIFO status.
TXTH	11:8	Transmit FIFO Threshold. These bits define the TX_FIFO interrupt threshold. Minimum FIFO length is 1, and maximum is 7.	
RFU	31:12	Reserved	N/A

13.10.11 I²C Transmit Register (I2CTX, Offset 0x0028)

Table 12-25. I2CTX (Offset 0x0028)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				TXFIFO[3:0]			
Reset	0000				0000			
Access	R				R			
Position	7	6	5	4	3	2	1	0
Field	RFU	RFU	RFU	RFU	RFU	RFU	TXLAST	TXRDY
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	RW	RW

Name	Bits	Description	Settings
TXRDY	0	Transmit FIFO Ready. When TXPRELD = 1: When entering TX FIFO Preload mode (TXPRELD = 1), the hardware clears TXRDY to 0. While TXRDY is still 0, the hardware sends a NACK when addressed for a slave transmit operation. This gives time for the software to preload data into the TX FIFO, initialize DMA, or both. Once ready, the software sets this bit to 1, and hardware then ACKs the next request for a transmit from the master. Once the transmission has begun, the hardware clears TXRDY. In this mode, TXRDY is also cleared when TX FIFO is flushed. When TXPRELD = 0: The hardware keeps TXRDY forced to 1. The hardware always ACKs requests for slave-transmit operations but clears the TX FIFO when addressed. This precludes preloading the TX FIFO, and guarantees that stale data intended for a previous transaction is not accidentally sent out for a new transaction.	
TXLAST	1	Transmit Last. This bit is used in slave mode only. (cleared by hardware)	0: Hold SCL low on TX_FIFO empty 1: End transaction on TX_FIFO empty
RFU	7:2	Reserved	N/A
TXFIFO	11:8	Transmit FIFO Count. These bits reflect the number of bytes in the TX_FIFO. These bits are read only and cleared to 0 when TX_FIFO is flushed and I2CEN=0.	
RFU	31:12	Reserved	N/A

13.10.12 I²C Data Register (I2CDATA, Offset 0x002C)

Table 12-26. I2CDATA (Offset 0x002C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	DATA[7:0]							
Reset	1111 1111							
Access	RW							

Name	Bits	Description	Settings
DATA	7:0	Data register. Data is read from or written to this location. Transmit and receive FIFO are separate, but both are addressed at this location. When IRXM=1, reads from this location and returns the data in the RX Shift register and does not affect the RX_FIFO. Reading from an empty RX FIFO returns 0xFF.	
RFU	31:8	Reserved	N/A

13.10.13 I²C Master Control Register (I2CMCN, Offset 0x0030)

Table 12-27. I2CMCN (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	SEA	RFU	RFU	RFU	RFU	STOP	RESTART	START
Reset	0	0	0	0	0	0	0	0
Access	RW	R	R	R	R	RW1S	RW1S	RW1S

Name	Bits	Description	Settings
START	0	Start. Setting this bit to 1 starts a master transfer. The bit is automatically cleared to 0 upon detection of a STOP at the end of a transaction with a master. This bit is set to 1 by the software. Use the hardware to clear the bit to 0. This bit is automatically cleared to 0 when MST=0, I2CEN=0, or upon arbitration loss.	
RESTART	1	Repeated START. Setting this bit to 1 generates a repeated START. The bit is automatically cleared to 0 once repeated START generation begins. This bit is set to 1 by the software. Use the hardware to clear the bit to 0. This bit is automatically cleared to 0 when MST=0, I2CEN=0, or upon arbitration loss.	
STOP	2	STOP. Setting this bit to 1 generates a STOP condition. The bit is automatically cleared to 0 once STOP generation begins. This bit is set to 1 by the software. Use the hardware to clear the bit to 0. This bit is automatically cleared to 0 when MST=0, I2CEN=0, or upon arbitration loss.	
RFU	6:3	Reserved	N/A
SEA	7	Slave Extend Address Select. This bit selects if the target slave addressing mode is 7-bit or 10-bit.	0: 7-bit address 1: 10-bit address
RFU	31:8	Reserved	N/A

13.10.14 I²C Clock Low Register (I2CCKL, Offset 0x0034)

Table 12-28. I2CCKL (Offset 0x0034)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[6:0]							CKL[8]
Reset	0000 000							0
Access	R							RW
Position	7	6	5	4	3	2	1	0
Field	CKL[7:0]							
Reset	0000 0001							
Access	RW							

Name	Bits	Description	Settings
CKL	8:0	Clock Low. In master mode, these bits define the SCL low period. $SCL\ low\ time = I2CCLKIN * (CKL + 1)$. Set CKL to a minimum value of 1. Writing any value less than 1 has the same effect as setting CKL=1. In slave mode, these bits define the time SCL is held low after data output.	0: Invalid 1: Minimum 511: Maximum
RFU	31:9	Reserved	N/A

13.10.15 I²C Clock High Register (I2CCKH, Offset 0x0038)

Table 12-29. I2CCKH (Offset 0x0038)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[6:0]							CKH[8]
Reset	0000 000							0
Access	R							RW
Position	7	6	5	4	3	2	1	0
Field	CKH[7:0]							
Reset	0000 0001							
Access	RW							

Name	Bits	Description	Settings
CKH	8:0	Clock High. In master mode, these bits define the SCL high period. $SCL\ High\ Time = I2CCLKIN * (CKH+1)$ Set CKH to a minimum value of 1. Writing any value less than 1 has the same effect as setting CKH=1.	0: Invalid 1: Minimum 511: Maximum
RFU	31:9	Reserved	N/A

13.10.16 I²C Timeout Register (I2CTO, Offset 0x0040)

Table 12-30. I2CTO (Offset 0x0040)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	TO[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	TO[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
TO	15:0	Timeout. Timeout = I2CCLKIN * (TO*8+3)	
RFU	31:16	Reserved	N/A

13.10.17 I²C Slave Address Register (I2CSLA, Offset 0x0044)

Table 12-31. I2CSLA (Offset 0x0044)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	EA	RFU[4:0]					SLA[9:8]	
Reset	0	000 00					00	
Access	RW	R					RW	
Position	7	6	5	4	3	2	1	0
Field	SLA[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
SLA	9:0	Slave Address	
RFU	14:10	Reserved	
EA	15	Extended Address Select. This bit selects if the SLA contains a 7-bit or 10-bit address.	0: 7-bit address 1: 10-bit address
RFU	31:16	Reserved	N/A

13.10.18 I²C DMA Register (I2CDMA, Offset 0x0048)

Table 12-32. I2CDMA (Offset 0x0048)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[5:0]						RXEN	TXEN
Reset	0000 00						0	0
Access	R						RW	RW

Name	Bits	Description	Settings
TXEN	0	TX Channel Enable	0: disabled 1: enabled
RXEN	1	RX Channel Enable	0: disabled 1: enabled
RFU	31:2	Reserved	N/A

13.10.19 I²C Deglitcher Register (I2CDGLTCHR, Offset 0x0000)

Table 12-33. I2CDGLTCHR (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[1:0]		I2C0DGEN1	I2C0DGEN0	RFU[3:0]			
Reset			0	0	0000			
Access			RW	RW	R			
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							

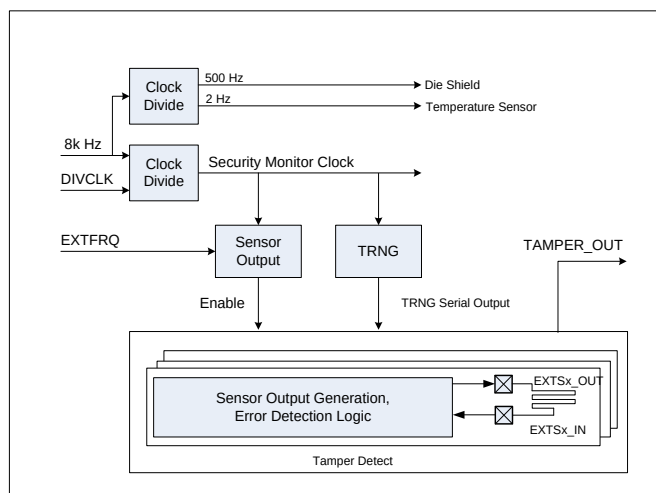
Name	Bits	Description	Settings
RFU	19:0	Reserved	N/A
I2C0DGEN0	20	I2C SDA Deglitcher Enable	0: Deglitcher disabled 1: Deglitcher enabled
I2C0DGEN1	21	I2C SCL Deglitcher Enable	0: Deglitcher disabled 1: Deglitcher enabled
RFU	31:22	Reserved	N/A

14.0 Security Monitor

The Security Monitor block is used to monitor system threat conditions. These include external sensor pairs, an internal die shield, temperature, and voltage monitors. An internal 8kHz clock provides time base for the security monitor block. This clock can be divided down by setting the Clock Divide bits (EXTSCN.DIVCLK). The resulting divided clock is used for all logic within the Security Monitor Block.

A dedicated Random Number Generator (TRNG) generates the security patterns for the external sensors and die shield.

Figure 14-1. Security Monitor Block Diagram



14.1 Destructive Reset Source

A Destructive Reset Source (DRS) can be triggered by any of the defined sources, including tamper detect inputs, environmental monitors if the lock bit (EXTSCN.LOCK) is set.

The DRS sources are:

- Die shield tamper
- Temperature out of range
- Voltage out of range
- External tamper
- Software initiated by setting SECALM.DRS

If the lock bit is set (EXTSCN.LOCK=1) DRS response initiates the following actions:

- It triggers Cortex-M4F NMI.
- AES secure key register is cleared.
- The DRS source flag(s) is (are) set.

Important Note: *If the lock bit is not set, the Alarm bit will be set upon a tamper event but keys will not be erased and no NMI or DRS will be triggered.*

The DRS source is recorded as flags in the SECALM and SECDIAG registers.

Upon detection of a tamper event, the secure monitor triggers the following actions:

1. Erase all three 256-bit keys, one AES and two PUF generated keys.
2. Assert the TAMPER_OUT pin to indicate a security breach has occurred.
3. If the tamper causes the die shield alarm, the PUF trim information is always destroyed.
4. If the tamper causes the external sensor alarm, the PUF trim information may also be destroyed. This feature is configurable by users through a register bit in the security monitor function. An OTP option is also implemented to NOT allow the PUF trim information to be destroyed by the external sensor alarm, overriding the user-selectable register bit.

The source flag in the SECALM register remains set until the software clears the flag. The source flag in the SECDIAG register remains set until a POR.

There is no priority in the DRS sources and the order in which they are listed have no impact on the operation of the DRS system. If multiple DRS sources happen simultaneously, or additional DRS sources are triggered while the device is in DRS reset, all applicable DRS source flags will be set. . The device is placed in DRS reset until the DRS source condition has cleared, or a minimum of 4 core clock cycles, whichever is longer.

14.1.1 External Sensor Test

To test external sensors without generating a DRS/NMI, which causes secret (Keys Storage Register) erasure, the user can enable the desired external sensors by using the EXT_S_EN bit with LOCK bit cleared. In this way, the external sensors are active, alarm status is logged in SECALM (can be cleared).

The DRS source is recorded as flags in the SECALM and SECDIAG registers.

The source flag in the SECALM register remains set until the software clears the flag. The source flag in the SECDIAG register remains set until a POR.

14.2 Active Die Shield

The die shield monitors the top metal shield. This network covers the whole chip area. The die shield is enabled by setting the Die Shield Enable (INTSCN.SHIELD_EN) bit to 1.

When enabled, a Destructive NMI (non-maskable interrupt) will be triggered upon physical tampering attack to the chip.

14.3 Temperature Sensor

The device temperature is monitored by a temperature sensor and is clocked by a 500 Hz clock.

If the device temperature is outside the allowable range, a DRS will be initiated. A DRS caused by the above temperature fault sets the corresponding flags in SECALM and SECDIAG registers. Once set, the SECALM flags are cleared by the software or a POR, whereas the SECDIAG flags are cleared by a POR only.

The temperature sensor can be disabled for power savings by clearing the Temperature Sensor Enable bit (INTSCN.TEMP_EN). While the temperature low threshold is adjustable, the high temperature threshold is set at +125°C.

14.4 Secure Voltage Monitor

The device incorporates both a high-voltage and low-voltage monitor during the BACKUP Mode.

The high-voltage monitor causes a DRS when the input voltage attached to the device exceeds the data sheet threshold.

The low-voltage monitor causes a DRS when the input voltage attached to the device is below the data sheet threshold.

A DRS caused by the above fault condition sets the corresponding flags in the SECALM and SEC DIAG registers. Once set, the SECALM flags are cleared by the software or a POR, whereas the SEC DIAG flags are cleared by a POR only.

Secure voltage monitor can be disabled by clearing the V_{DD} Monitor Enable bit (INTSCN.V_{DD}_MON_EN).

14.5 External Sensor

There is one pair of External Sensor in this device. Each sensor pair consists of an output (EXTS_x_OUT) and input (EXTS_x_IN). Using the dedicate TRNG, each sensor output sends a unique random digital signal, which should be fed to an external mesh or contact point and then back to the corresponding sensor input. As long as the mesh remains intact (or the contact remains closed), the signal on the input will be filtered to remove transient glitches and compared to the output signal.

The sensor input sample frequency is 8kHz (default frequency). The rate of change of the sensor output is programmable from 2kHz down to 31Hz with the default input clock. This is accomplished by dividing the 8kHz clock by a user-defined value (DIVCLK, EXTFRQ) to produce a very slow local clock that is used to switch the sensor outputs. After changing sample frequency or sensor output frequency (DIVCLK, EXTFRQ), there may be a transient time up to about 100 clock cycles when the sensor is activated.

An error counter is used for glitch filtering purposes. At each mismatch between EXTS_x_IN and EXTS_x_OUT, the error counter is incremented. The counter is automatically reset each sensor bit period. If the counter exceeds its programmed threshold (EXTCNT), an alarm signal is raised, the status register is updated and a DRS is fired.

To determine the EXTCNT value:

Sample Frequency,		$f_{\text{SAMPLE}} = 8\text{kHz} / \text{Clock Divide Factor (DIVCLK)}$
Sensor Output Frequency,	f_{OUT}	$= f_{\text{SAMPLE}} / \text{Output Frequency Factor (EXTFRQ)}$
Glitch Filter Width (in second),	GFW	$= (\text{EXTCNT} + 1) * 1/f_{\text{SAMPLE}} \Rightarrow \text{EXTCNT} = \text{GFW} * f_{\text{SAMPLE}} - 1$

Note that $\text{GFW} < 1/f_{\text{OUT}}$. Otherwise, the tamper is not detected.

Therefore, $\text{EXTCNT} < (f_{\text{SAMPLE}}/f_{\text{OUT}}) - 1$

Example 1:

DIVCLK=000b

EXTFRQ=000b

$f_{\text{SAMPLE}} = 8\text{kHz}/1 = 8\text{kHz}$

$f_{\text{OUT}} = 8\text{kHz}/4 = 2\text{kHz}$

$\text{EXTCNT} < 8\text{kHz}/2\text{kHz} - 1 = 3$

Therefore, set EXTCNT from 0 to 2.

Maximum GFW < 1/2kHz (0.5ms)

To filter a 0.2ms glitch

GFW = 0.2 ms

$\text{EXTCNT} = 0.2 \text{ ms} * 8\text{kHz} - 1 = 0.6$

Therefore, set EXTCNT to 0.

Example 2:

DIVCLK=001b
EXTFRQ=010b

$f_{\text{SAMPLE}} = 8\text{kHz}/2 = 4\text{kHz}$
 $f_{\text{OUT}} = 4\text{kHz}/16 = 250\text{Hz}$

$\text{EXTCNT} < 4\text{kHz}/250\text{Hz} - 1 = 15$.
Therefore, set EXTCNT from 0 to 14.
Maximum GFW < 1/250Hz (4ms)

To filter a 1ms glitch
GFW = 1 ms

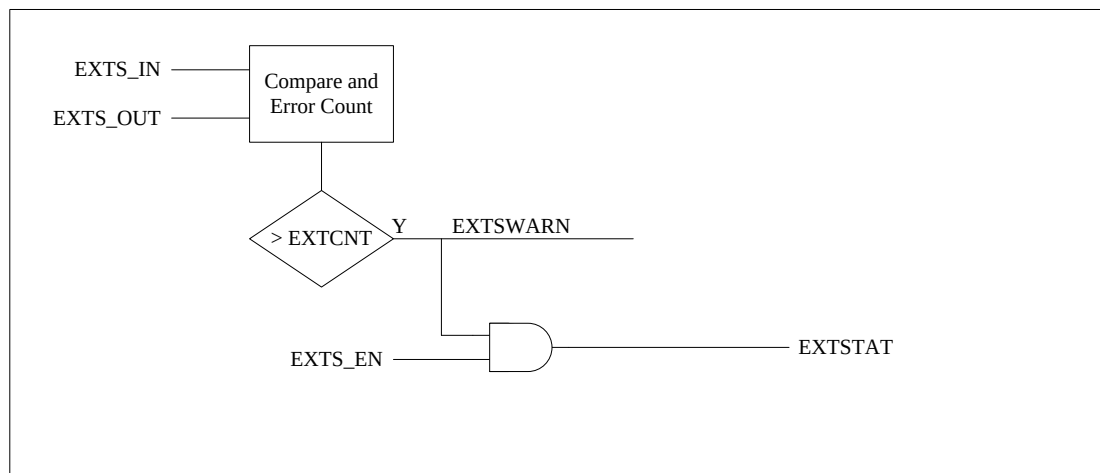
$\text{EXTCNT} = 1 \text{ ms} * 4\text{kHz} - 1 = 3$
Therefore, set EXTCNT from 0 to 2

14.5.1 External Sensor Warning

On initial power up, EXTSx_IN and EXTSx_OUT may not necessarily be the same. To prevent enabling external sensor tamper detect prematurely, software should check the status of each pair before enabling it. To check the status, software can read the External Sensor Warning (SECALM.EXTSWARN). The EXTSWARN bit is set to 1 when the error count exceeds the programmed threshold. It remains set until the software clears it and the error count no longer exceeds the programmed threshold. If EXTSWARN=1, then the sensor input and output has not settled to the same initial condition yet.

This is shown in Figure 14-2. For an enabled sensor, if the error count exceeds programmed threshold, the corresponding External Sensor Interrupt (EXTSI) and the External Sensor Tamper Detect (EXTSTAT) flags are set to 1. When any of the EXTSTAT flags are set, a DRS is generated, causing a non-maskable interrupt.

Figure 14-2. External Sensors Block Diagram



14.5.2 Power Fail Warning

The interrupt fires when any of the supplies are out of range. For location of the power fail warning interrupt vector, refer to Interrupt vector table on page 66. The power fail warning sources are:

- V_{DD} over voltage
- V_{DD} under voltage

14.6 Security Monitor Registers

Address assignments for the Security Monitor registers are outlined in Table 14-1.

Table 14-1. Security Monitor Register Address (Base ADDR = 0x4000_4000)

Offset	Access	Register	Description
0x0000	RW	EXTSCN	External Sensor Control Register
0x0004	RW	INTSCN	Internal Sensor Control Register
0x0008	RW	SECALM	Security Alarm Register
0x000C	R	SECDIAG	Security Diagnostic Register
0x0010 ... 0x0030	R	RFU	Reserved
0x0034	R	SECST	Security Monitor Status Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

14.6.1 External Sensor Control Register (EXTSCN, Offset 0x0000)

This register is retained in the BACKUP Mode. This register can only be written to when LOCK=0.

Table 14-2. EXTSCN (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	LOCK	BUSY	RFU[2:0]			DIVCLK[2:0]		
POR	0	0	000			000		
Other Reset	N/A	N/A	000			N/A		
Access	RS	R	R			RW		

Position	23	22	21	20	19	18	17	16
Field	EXTFRQ[2:0]			EXTCNT[4:0]				
POR	000			0 0000				
Other Reset	N/A			N/A				
Access	RW			RW				

Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
POR	0000 0000							
Other Reset	0000 0000							
Access	R							

Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							EXTS_EN
POR	0000 000							0
Other Reset	0000 000							N/A
Access	R							RW

Name	Bits	Description	Settings
EXTS_EN	0	External Sensor Enable. Each bit enables a corresponding external sensor input/output pair.	0: External sensor disabled 1: External sensor enabled
RFU	15:1	Reserved	N/A
EXTCNT	20:16	External Sensor Error Counter. These bits set the number of external sensors accepted mismatches that must occur within a single bit period before an external sensor alarm is triggered.	Number of accepted mismatches = EXTCNT
EXTFRQ	23:21	External Sensor Frequency. These bits define the frequency at which the external sensors are clocked to/from the EXTS_IN and EXTS_OUT pair.	000: /4 (2000Hz) 001: /8 (1000 Hz) 010: /16 (500 Hz) 011: /32 (250 Hz) 100: /64 (125 Hz) 101: /128 (63 Hz) 110: /256 (31 Hz) 111: Reserved * The above table assumes an undivided 8kHz reference clock.
DIVCLK	26:24	Clock Divide. These bits are used to divide the 8kHz input clock. The resulting divided clock is used for all logic within the Security Monitor Block. Note: If the input clock is divided with these bits, the error count threshold table and output frequency are affected accordingly with the same divide factor.	000: /1 (8000 Hz) 001: /2 (4000 Hz) 010: /4 (2000 Hz) 011: /8 (1000 Hz) 100: /16 (500 Hz) 101: /32 (250 Hz) 110: /64 (125 Hz) 111: Reserved

Name	Bits	Description	Settings
RFU	29:27	Reserved	N/A
BUSY	30	Busy. This bit is set to 1 by the hardware after the EXTSCN register is written to. This bit is automatically cleared to 0 after this register information is transferred to the security monitor domain.	0: Idle 1: Update in progress
LOCK	31	Lock Register. Once locked, the EXTSCN register can no longer be modified. Only a POR clears this bit. DRS/NMI is only generated when this bit is set.	0: Not locked. Writes to this register allowed. 1: Locked. Writes to this register ignored.

14.6.2 Internal Sensor Control Register (INTSCN, Offset 0x0004)

This register is retained in the BACKUP Mode. This register can only be written to when LOCK=0.

Table 14-3. INTSCN (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	LOCK	RFU[6:0]						
POR	0	000 0000						
Other Reset	N/A	000 0000						
Access	RS	R						

Position	23	22	21	20	19	18	17	16
Field	RFU[3:0]				VTM_LOTHSEL [1:0]		RFU	LOTEMP_SEL
POR	0000				00		0	0
Other Reset	0000				00		N/A	N/A
Access	R				RW		R	RW

Position	15	14	13	12	11	10	9	8
Field	RFU[4:0]					PUF_TRM_ERASE_EN	RFU	DFD_SYDBY_EN
POR	0000 0					0	0	0
Other Reset	0000 0					0	0	0
Access	R					RW	R	RW

Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			DFD_NMI	DFD_EN	VDD_MON_EN	TEMP_EN	SHIELD_EN
POR	000			0	0	0	0	0
Other Reset	000			0	0	N/A	N/A	N/A
Access	RW			RW	RW	RW	RW	RW

Name	Bits	Description	Settings
SHIELD_EN	0	Die Shield Enable.	0: Die shield sensors powered off 1: Die shield sensors enabled
TEMP_EN	1	Temperature Sensor Enable.	0: Temperature sensor powered off 1: Temperature sensor enabled
VDD_MON_EN	2	VDD Monitor Enable.	0: VDD voltage monitor powered off 1: VDD voltage monitor enabled
DFD_EN	3	Digital Fault Detector Enable. This flag is set if there is a glitch on logic	0: DFD enabled 1: DFD disabled
DFD_NMI	4	Digital Fault Detector Non-maskable Interrupt. If this bit is set, DRS/NMI is generated upon DFD alarm. Otherwise PFW IRQ is generated.	0: DRS/NMI 1: PFW IRQ
RFU	7:5	Reserved.	N/A
DFD_STDBY_EN	8	Set this bit to disable DFD sensor during Low Power Mode	0: DFD enabled during Low Power Mode 1: DFD disabled during Low Power Mode
RFU	9	Reserved	N/A
PUF_TRM_ERASE_EN	10	If set, erase PUF trim data upon External Sensor or Die Shield detection. Otherwise PUF trim is only erased upon Die Shield detection.	0: PUF trim erased only upon Die Shield alarm 1: PUF trim erased upon either Die Shield or External sensor alarm
RFU	15:11	Reserved	N/A
LOTEMP_SEL	16	Low Temperature Detection Select.	0: ~ -50°C 1: ~ -30°C

Name	Bits	Description	Settings
RFU	17	Reserved.	N/A
VTM_LOTHSEL	19:18	VTM low threshold detection	00: 1.6V 01: 2.2V 1x: 2.8V
RFU	30:20	Reserved	N/A
LOCK	31	Lock Register. Once locked, the INTSCN register can no longer be modified. Only a POR clears this bit.	0: Not locked. Writes to this register allowed. 1: Locked. Writes to this register ignored.

14.6.3 Security Alarm Register (SECALM, Offset 0x0008)

This register is retained in the BACKUP Mode. This register bits are set by hardware on detection of a DRS and can be cleared by the software.

Table 14-4. SECALM (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[5:0]							EXTSWARN
POR	0000 000							0
Other Reset	N/A							N/A
Access	R							RC
Position	23	22	21	20	19	18	17	16
Field	RFU[5:0]							EXTSTAT
POR	0000 000							0
Other Reset	N/A							N/A
Access	R							RC
Position	15	14	13	12	11	10	9	8
Field	RFU[5:0]						RFU	DFD
POR	0000 00						0	0
Other Reset	N/A						0	0
Access	R						R	RC
Position	7	6	5	4	3	2	1	0
Field	EXTF	SVDDHI	SVDDLO	HITEMP	LOTEMP	SHIELDF	KEYWIPE	DRS
POR	0	0	0	0	0	0	0	0
Other Reset	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Access	RC	RC	RC	RC	RC	RC	RW	RW

Name	Bits	Description	Settings
DRS	0	Destructive Reset Trigger. Setting this bit generates a DRS. This bit is self-cleared by the hardware.	0: No effect. 1: Set to 1 to initiate a DRS.
KEYWIPE	1	Key Wipe Trigger. Set to 1 to initiate a wipe of the Keys Storage Register. It does not reset the part. AES and DES registers are not affected by this bit. This bit is automatically cleared to 0 after the keys have been wiped	0: No effect. 1: Initiate key wipe
SHIELDF	2	Die Shield Flag.	0: No fault detected. 1: Die shield fault detected.
LOTEMP	3	Low Temperature Detect.	0: No fault detected. 1: Device temperature below low temperature threshold detected.
HITEMP	4	High Temperature Detect.	0: No fault detected. 1: Device temperature above high temperature threshold detected.
SVDDLO	5	Undervoltage tamper Detect	0: No fault detected. 1: Under voltage detected.
SVDDHI	6	Overvoltage tamper Detect	0: No fault detected. 1: Over voltage detected.
EXTF	7	External Sensor Flag. This bit is set to 1 when any of the EXTSTAT bits are set.	0: No fault detected. 1: External tamper detected.

Name	Bits	Description	Settings
DFD	8	Digital Fault Detector	0: No fault detected. 1: Digital Fault detected.
RFU	9	VMAIN Power Fail Fla	0: No fault detected. 1: VMAIN under voltage detected.
RFU	15:10	Reserved	N/A
EXTSTAT	16	External Sensor Detect. There is one external sensor input/output pair. The tamper detect is only active when it is enabled. This bit needs to be cleared in the software after a tamper event to re-arm the sensor.	0: No fault detected. 1: External tamper detected.
RFU	23:17	Reserved	N/A
EXTSWARN	24	External Sensor Warning Ready flag. There is one external sensor. The tamper detect warning flag is set, regardless of the external sensor being enabled.	0: No fault detected 1: External tamper warning.
RFU	31:25	Reserved	N/A

14.6.4 Security Diagnostic Register (SECDIAG, Offset 0x000C)

This register is available in backed-up. This register bits are set by the hardware on the detection of a DRS and can be cleared by POR only.

Table 14-5. SECDIAG (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
POR	0000 0000							
Other Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[6:0]							EXTSTAT
POR	0000 000							0
Other Reset	0000 000							N/A
Access	R							R
Position	15	14	13	12	11	10	9	8
Field	DFDF	RFU		BBKEY2_ZERO	BBKEY1_ZERO	BBKEY0_ZERO	AESKT	RFU
POR	0	00		0	0	0	0	0
Other Reset	N/A	00		N/A	N/A	N/A	N/A	0
Access	R	R		R	R	R	R	R
Position	7	6	5	4	3	2	1	0
Field	DYNF	SVDDHI	SVDDLO	HITEMP	LOTEMP	SHIELDF	RFU	PORF
POR	0	0	0	0	0	0	0	1
Other Reset	N/A	N/A	N/A	N/A	N/A	N/A	0	N/A
Access	R	R	R	R	R	R	R	W1C

Name	Bits	Description	Settings
PORF	0	Power-On-Reset Flag. This bit is set once the power supply is connected. Write 1 to clear this bit.	0: POR has not occurred 1: POR has occurred
RFU	1	Reserved	N/A
SHIELDF	2	Die Shield Flag.	0: No fault detected. 1: Die shield fault detected.
LOTEMP	3	Low Temperature Detect	0: No fault detected. 1: Device temperature below low temperature threshold detected.
HITEMP	4	High Temperature Detect	0: No fault detected. 1: Device temperature above high temperature threshold detected.
SVDDLO	5	Undervoltage tamper Detect	0: No fault detected. 1: Undervoltage detected.
SVDDHI	6	Overvoltage tamper Detect	0: No fault detected. 1: Overvoltage detected.
DYNF	7	Dynamic Sensor Flag. This bit is set to 1 when any of the EXTSTAT bits are set.	0: No fault detected. 1: External tamper detected.
RFU	8	Reserved	N/A
AESKT	9	AES Key Transfer to Secure Key Storage Register. This bit is set to 1 when AES Key has been transferred from TRNG to the Keys Storage Register. This bit can be reset by a POR, DRS, KEYWIPE or DRS bit in SECALM.	0: Key has not been transferred or has been erased by DRS 1: Key has been transferred.

Name	Bits	Description	Settings
KEY0_ZERO	10	Key0 is cleared.	0: Key is set 1: Key is cleared
KEY1_ZERO	11	Key1 is cleared.	0: Key is set 1: Key is cleared
KEY2_ZERO	12	Key2 is cleared.	0: Key is set 1: Key is cleared
RFU	14:13	Reserved	N/A
DFDF	15	Digital Fault Detector Flag	0: No fault detected 1: Fault detected
EXTSTAT	16	External Sensor Detect. Each bit corresponds to one external sensor input/output pair.	0: No fault detected. 1: External tamper detected.
RFU	31:17	Reserved	N/A

14.6.5 Security Monitor Status Register (SECST, Offset 0x0034)

The register reflects the availability of security monitor registers. When the corresponding bit is 0, the associated register has been updated and access (read/write operation) is authorized. When the corresponding bit is 1, the associated register is being updated and access is not authorized. This register should be polled after each write to a security monitor register to avoid reading back the previous value.

Table 14-6. SECST (Offset 0x0034)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[4:0]					SECALRS	INTSRS	EXTSRS
Reset	0000 0					0	0	0
Access	R					R	R	R

Name	Bits	Description	Settings
EXTSRS	0	External Sensor Control Register Status	0:Access authorized 1:Access not authorized.
INTSRS	1	Internal Sensor Control Register Status	0:Access authorized 1:Access not authorized.
SECALRS	2	Security Alarm Register Status	0:Access authorized 1:Access not authorized.
RFU	31:3	Reserved	N/A

15.0 Quad Serial Peripheral Interface (QSPI)

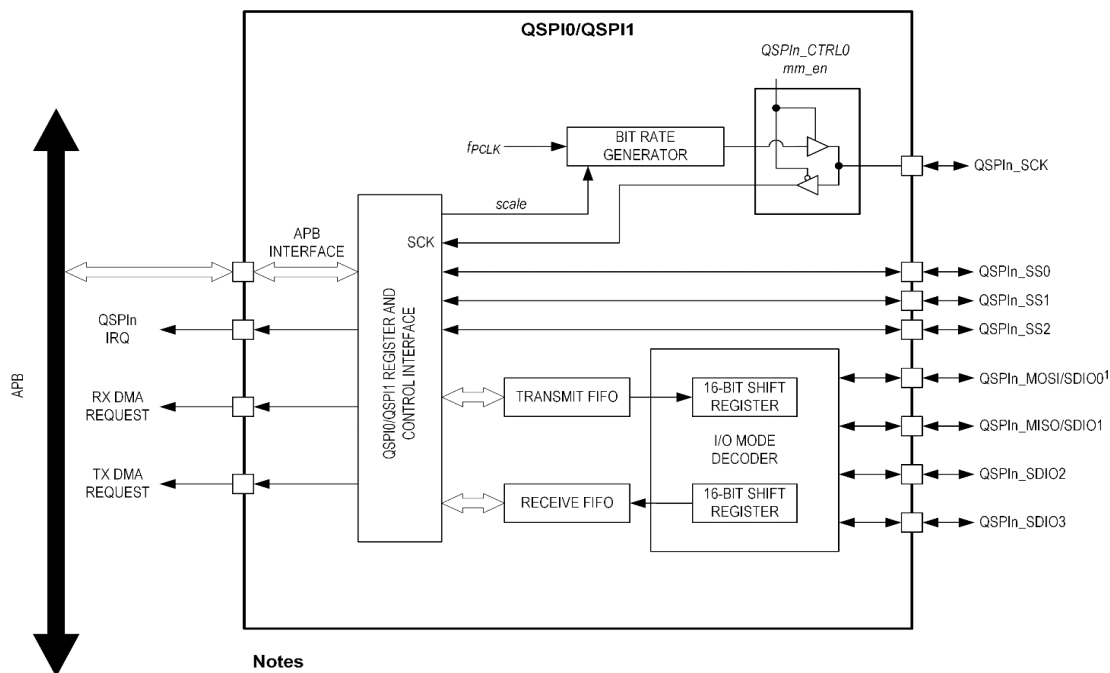
The Quad Serial Peripheral Interface (QSPI) is a highly configurable, flexible, and efficient synchronous interface between multiple SPI devices on a single bus. The SPI bus uses a single clock signal, single, dual, or quad data lines, and one or more slave select lines for communication with external SPI devices. The QSPI ports support full-duplex, bi-direction I/O and each QSPI includes a Bit Rate Generator (BRG) for generating the clock signal when operating in master mode. Each QSPI port operates independently and requires minimal processor overhead.

This device provides one instance of the QSPI and one regular SPI interface. The features of the SPI include:

- Dedicated Bit Rate Generator for precision serial clock generation in Master Mode
 - ♦ Up to $\frac{f_{PCLK}}{2}$ for QSPI0 and QSPI1
 - ♦ Up to $\frac{f_{HCLK}}{2}$ for QSPI2
 - ♦ Programmable SCK duty cycle timing
- Full-duplex, synchronous communication of 1 to 16-bit characters
- 3-wire and 4-wire SPI operation for single-bit communication
- Single, Dual and Quad I/O
- Byte-wide Transmit and Receive FIFOs with 32-byte depth
 - ♦ For character sizes greater than 8, each character uses 2 entries per character resulting in 16 entries for the Transmit and Receive FIFO
- Transmit and Receive DMA support
- SPI modes 0, 1, 2, 3
- Three slave select lines for each QSPI port
 - ♦ Programmable slave select level
 - ♦ Programmable slave select timing with respect to SCK starting edge and ending edge
- Multi-master mode fault detection

Figure 15-1 is a high-level block diagram for QSPI interface. An SPI network uses a single master and one or more slaves for any given transaction. The QSPI peripherals support single master and multi-master SPI networks.

Figure 15-1. QSPI Block Diagram



15.1 Four-Wire SPI Signals

SPI devices operate as either a master or slave device. In four-wire SPI, four signals are required for communication as shown in Table 15-1

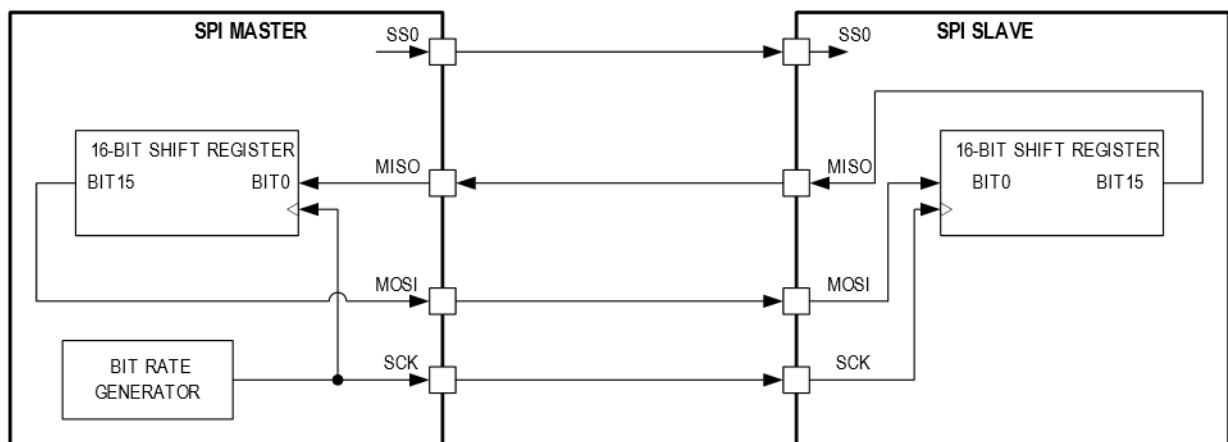
Table 15-1. Four-Wire SPI Signal

Signal	Description	Direction
SCK	Serial Clock	The master generates the Serial Clock signal, which is an output from the master and an input to the slave.
MOSI	Master Output Slave Input	In master mode, this signal is used as an output for sending data to the slave. In slave mode, this is the input data from the master.
MISO	Master Input Slave Output	In master mode, this signal is used as an input for receiving data from the slave. In slave mode, this signal is an output for transmitting data to the master.
SS	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. In slave mode, this signal is an input used to indicate the master is going to start communication.

This device supports two slave select lines on QSPI0 and four on SPI1.

A typical four-wire SPI network is shown in Figure 15-2. In a typical SPI network, the master device selects the slave device using the slave select output. The master starts the communication by selecting the slave device by asserting the slave select output. The master then starts the SPI clock via the SCK output pin. When a slave device's slave select pin is deasserted, the device is required to put the SPI pins in tri-state mode.

Figure 15-2. 4-Wire SPI Connection Diagram



15.2 Three-Wire SPI Signal

Each QSPI port supports three-wire SPI. In three-wire SPI operation, the MOSI signal is used as a bi-directional, half-duplex I/O referred to as Slave Input Slave Output (SISO). Three-wire SPI also uses a serial clock signal generated by the master and a slave select pin controlled by the master. Table 15-2 describes each of the signals used in three-wire SPI communication.

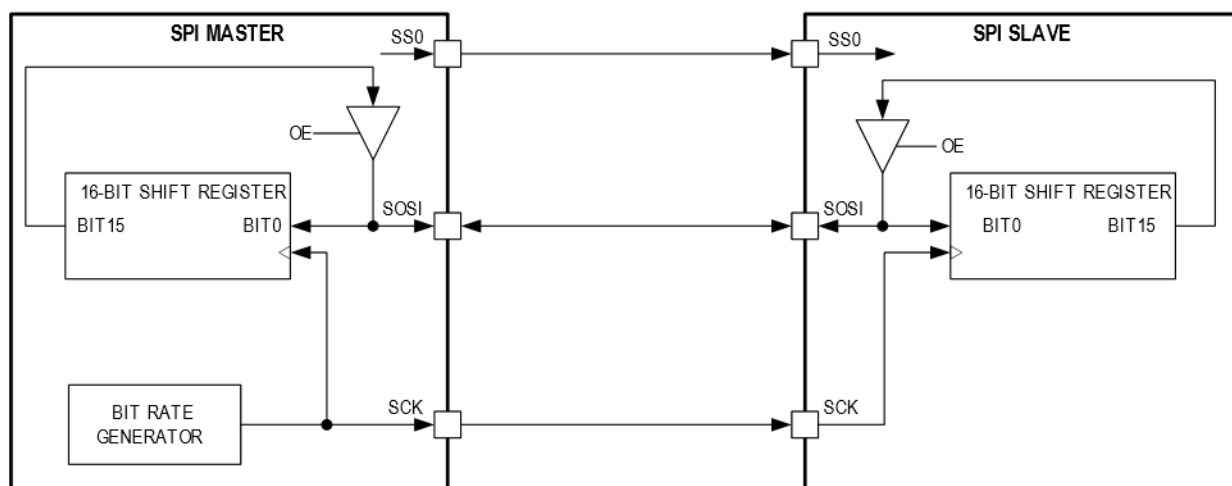
A three-wire SPI network is shown in Figure 15-3. The master device selects the slave device using the slave select output. The communication starts with the master asserting the slave select line and then starting the clock (SCK). In

three wire SPI communication, the master and slave must both know the intended direction of the data to prevent bus contention. For a write, the master drives the data out the SISO pin. For a read, the master must release the SISO line and let the slave drive the SISO line, usually on the second edge of a clock cycle. The direction of transmission is controlled using the FIFO. Writing to the FIFO starts the three-wire SPI write and reading from the FIFO starts a three-wire SPI read transaction.

Table 15-2. Three-Wire SPI Signals

Signal	Description	Direction
SCK	Serial Clock	The master generates the Serial Clock signal, which is an output from the master and an input to the slave.
SISO	Slave Input Slave Output	This is a half-duplex, bidirectional I/O pin used for communication between the SPI master and slave. This signal is used to transmit data from the master to the slave and to receive data from the slave by the master.
SS	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. In slave mode, this signal is an input used to indicate an external SPI master is starting communication.

Figure 15-3. Generic 3-Wire SPI Master to Slave Connection



15.3 Pin Configuration

Pin selection and configuration is required to use the QSPI port. The following information applies to SPI master and slave operation as well as three-wire, four-wire, dual, and quad-mode communications. Refer to the part data sheet for specific pin mapping of QSPI.

When the QSPI port is disabled, `QSPI_CTRL0.enable = 0`, the GPIO pins enabled for QSPI alternate function are placed in high-impedance input mode.

15.3.1 Four-wire SPI Pin Configuration

Four-wire SPI uses SCK, MISO, MOSI, and one or more SS pins. Four-wire SPI may use more than one slave select pin for a transaction, resulting in more than four wires. However, the communication is referred to as four wire for legacy reasons.

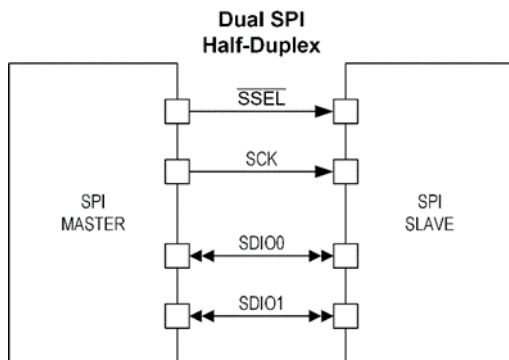
15.3.2 Three-wire SPI Pin Configuration

Three-wire SPI uses SCK, MOSI (SOSI), and one or more slave select pins for an SPI transaction. Three-wire SPI configuration is identical to the four-wire configuration except the QSPI_MISO does not need a set up for the QSPI alternate function. The direction of communication in three-wire SPI mode is controlled by the QSPI Transmit and Receive FIFO enables. Enabling the Receive FIFO and disabling the Transmit FIFO indicates a read transaction. Enabling the Transmit FIFO and disabling the Receive FIFO indicates a write transaction. It is an illegal condition to enable both the Transmit and Receive FIFOs in a three-wire SPI operation.

15.3.3 Dual-Mode SPI Pin Configuration

In Dual mode SPI two I/O pins are used to transmit 2-bits of data per SCK clock cycle. The communication is half-duplex and the direction of the data transmission must be known by both the master and slave for a given transaction. Dual-mode SPI uses SCK, SDIO0, SDIO1, and one or more slave select lines as shown in Figure 15-4. The configuration of the GPIO pins for dual-mode SPI is identical to four-wire SPI, and the mode is controlled by setting QSPI_CTRL2.data_width to 1 indicating to the QSPI hardware to use SDIO0 and SDIO1 for half-duplex communication rather than full-duplex communication.

Figure 15-4. Dual Mode SPI Connection Diagram



15.3.4 Quad-Mode SPI Pin Configuration

Quad-mode SPI uses four I/O pins to transmit four-bits of data per transaction. In Quad mode SPI, the communication is half-duplex and the master and slave must know the direction of transmission for each transaction. Quad-mode SPI uses SCK, SDIO0, SDIO1, SDIO2, SDIO3, and one or more slave select pins. Configure the QSPIn pins for quad-mode SPI by following the steps for four-wire SPI with the additional steps below to select the SDIO2 and SDIO3 alternate functions.

Quad-mode SPI transmits four bits per SCK cycle. The quad-mode SPI is selected by setting QSPI_CTRL2.data_width to 2.

15.4 QSPI Clock Configuration

15.4.1 Serial Clock

The SCK signal synchronizes data movement in and out of the device. The master drives SCK as an output to the slave's SCK pin. When QSPI is set to master mode, the QSPI bit rate generator creates the serial clock and outputs it on the configured QSPI_SCK pin. When QSPI is configured for slave operation the QSPI_SCK pin is an input from the external master and the QSPI hardware synchronizes communications using the SCK input. Operating as a slave, if a QSPI slave select input is not asserted, the QSPI ignores any signals on the serial clock and serial data lines.

When QSPI is configured for slave operation, the maximum SCK input frequency supported is $f_{SCK}=f_{PCLK}/8$. For example, if $f_{PCLK}=48\text{MHz}$, the maximum SPI clock frequency supported in the slave mode for QSPI is 6MHz.

In both master and slave devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Data availability and sampling time is controlled using the SPI phase control field, QSPI_CTRL2.phase. The SCK clock polarity field, QSPI_CTRL.clkpol, controls if the SCK signal is active high or active low.

The QSPI peripheral supports four combinations of SCK phase and polarity referred to as SPI Modes 0, 1, 2, and 3. Clock Polarity (QSPI_CTRL2.clkpol) selects an active low/high clock and has no effect on the transfer format. Clock Phase (QSPI_CTRL2.phase) selects one of two different transfer formats.

For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCK edge for the slave to latch the data. Refer to section 15.4.4 Clock Phase and Polarity Control for additional details.

15.4.2 SPI Peripheral Clock

The System Peripheral Clock, PCLK, drives the QSPI peripheral clock. The SPI0 provides an internal clock, SPI0_CLK, that is used within the SPI peripheral for the base clock to control the module and generate the SCK clock when in master mode. Set the SPI0 internal clock using the field QSPI_CLK_CFG.scale as shown in the following equation. The valid settings for QSPI_CLK_CFG.scale are 0 to 8, allowing a divisor of 1 to 256.

$$f_{\text{SPI\_CLK}} = \frac{f_{\text{PCLK}}}{2^{\text{SCALE}}}$$

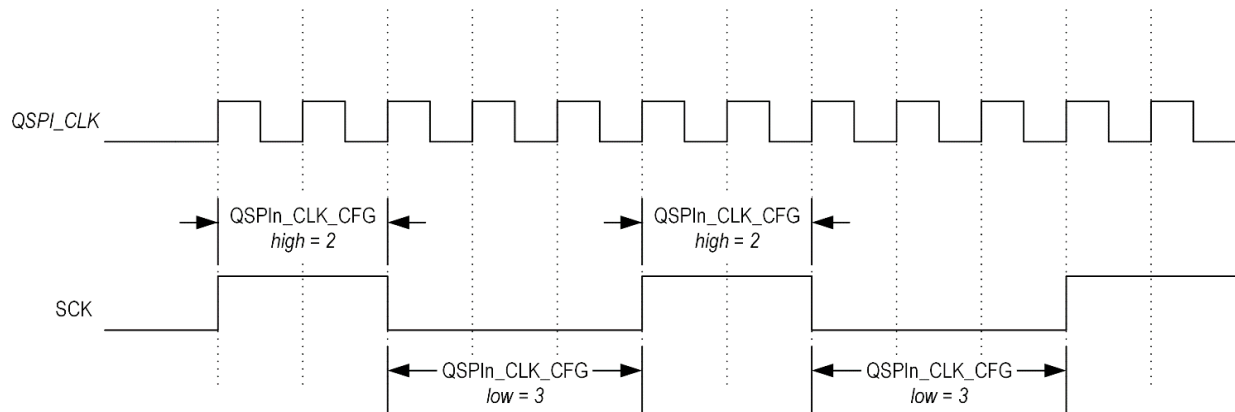
15.4.3 Master-Mode Serial Clock Generation

In master and multi-master mode the SCK clock is generated by the master. The SPI0 provides control for both the high time and low time of the SCK clock. This control allows setting the high and low times for the SCK to duty cycles other than 50%, if required. The SCK clock uses the QSPI peripheral clock as a base value and the high and low values are a count of the number of $f_{\text{SPI\_CLK}}$ clocks. Figure 15-5 visually represents the use of the QSPI_CLK_CFG.hi and QSPI_CLK_CFG.lo fields for a non-50% duty cycle serial clock generation. Refer to the following two equations to calculate the SCK high and low time from the QSPI_CLK_CFG.hi and QSPI_CLK_CFG.lo field values.

$$t_{\text{SCK\_HI}} = t_{\text{QSPI\_CLK}} \times \text{QSPI\_CLK\_CFG.hi}$$

$$t_{\text{SCK\_LOW}} = t_{\text{QSPI\_CLK}} \times \text{QSPI\_CLK\_CFG.lo}$$

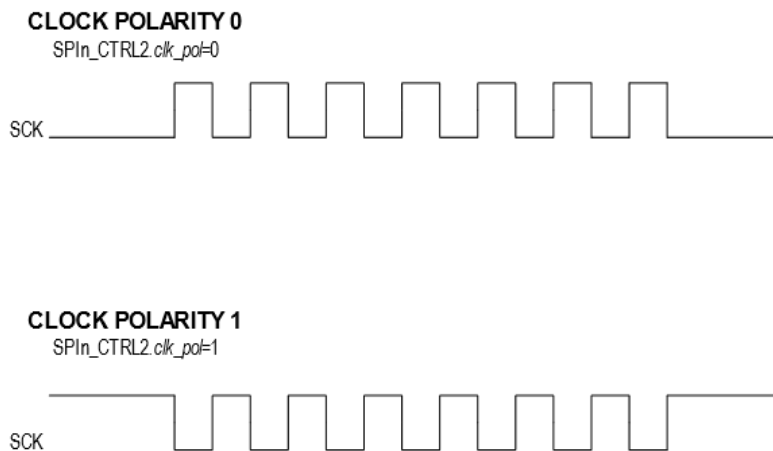
Figure 15-5. SCK Clock Rate Control



15.4.4 Clock Phase and Polarity Control

QSPI supports four combinations of clock and phase polarity as shown in Table 15-3 . Clock polarity is controlled using the bit QSPI_CTRL2.clkpol and determines if the clock is active high or active low as shown in Figure 15-6. Clock polarity does not affect the transfer format for SPI. Clock phase determines when the data must be stable for sampling. Setting the clock phase to 0, QSPI_CTRL2.phase = 0, dictates the SPI data is sampled on the initial SPI clock edge regardless of clock polarity. Phase 1, QSPI_CTRL2.phase = 1, results in data sample occurring on the second edge of the clock regardless of clock polarity.

Figure 15-6. SPI Clock Polarity



For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCK edge for the slave to latch the data.

Table 15-3. SPI Mode Clock Phase and Polarity Operation

SPI Mode	QSPI_CTRL2 PHASE	QSPI_CTRL2 CLKPOL	SCLK Transmit Edge	SCLK Receive Edge	SCLK Idle State
0	0	0	Falling	Rising	Low
1	0	1	Rising	Falling	High
2	1	0	Rising	Falling	Low
3	1	1	Falling	Rising	High

15.4.5 QSPI FIFO

The Transmit FIFO hardware is 32 bytes deep. The write data width can be 8-, 16- or 32-bits wide. A 16-bit write queues a 16-bit word to the FIFO hardware. A 32-bit write queues two 16-bit words to the FIFO hardware with the least significant word dequeued first. Bytes must be written to two consecutive byte addresses, with the odd byte as the most significant byte and the even byte as the least significant byte. The FIFO logic waits for both the odd and even bytes to be written to this register space before dequeuing the 16-bit result to the FIFO.

The Receive FIFO hardware is 32 bytes deep. Read data width can be 8-, 16- or 32-bits. A byte read from this register dequeues one byte from the FIFO. A 16-bit read from this register dequeues two bytes from the FIFO, least significant byte first. A 32-bit read from this register dequeues four bytes from the FIFO, least significant byte first.

15.4.6 QSPI Interrupt and Wakeups

The QSPI supports multiple interrupt sources. Status flags for each interrupt are set regardless of the state of the interrupt enable bit for that event. The event happens once when the condition is satisfied. The status flag must be cleared by the firmware by writing a 1 to the interrupt flag.

The following FIFO interrupts are supported:

- Transmit FIFO Empty
- Transmit FIFO Threshold
- Receive FIFO Full
- Receive FIFO threshold
- Transmit FIFO Underrun
 - Slave mode only, master mode stalls the serial clock
- Transmit FIFO Overrun
- Receive FIFO Underrun
- Receive FIFO Overrun (Slave Mode only, Master Mode stalls the clock)

QSPI supports interrupts for the internal state of the QSPI as well as external signals. The following transmission interrupts are supported:

- SSn asserted or deasserted
- SPI transaction complete
- Slave mode transaction aborted
- Multi-master fault

The QSPI port can wake up the microcontroller for low-power modes when the wake event is enabled. QSPI events that can wake the microcontroller are:

- Receive FIFO full
- Transmit FIFO empty
- Receive FIFO threshold
- Transmit FIFO threshold

15.5 SPI Registers

Address assignments for SPI registers within each block are outlined in Table 2-2. Peripheral Bus Region. Reserved register bits should only be written as 0.

Each SPI instance is controlled by a block of registers assigned to that port; all blocks contain identical control, interrupt, status, and FIFO read/write registers.

Register names for different instances can be defined by appending the instance number to the peripheral name. So, for instance, the DMA Register for SPI 0 would be QSPI0_DMA, while the Control Register for SPI 1 would be QSPI1_DMA, and so on.

Table 15-4. SPI Register Addresses (for each instance)

Offset	Access	Register	Description
0x0000	RW	QSPI_DATA	QSPI FIFO Data Register
0x0004	RW	QSPI_MSTR_CNTL	QSPI Master Signals Control Register
0x0008	RW	QSPI_TRNMT_SIZE	QSPI Transmit Packet Size Register
0x000C	RW	QSPI_STATIC_CONFIG	QSPI Static Configuration Register
0x0010	RW	QSPI_SS_TIME	QSPI Slave Select Timing Register
0x0014	RW	QSPI_CLK_CFG	QSPI Master Clock Configuration Register
0x0018	R	RFU	Reserved
0x001C	RW	QSPI_DMA	QSPI DMA Control Register
0x0020	RW1C	QSPI_INT_FL	QSPI Interrupt Flag Register
0x0024	RW	QSPI_INT_EN	QSPI Interrupt Enable Register
0x0028	RW1C	QSPI_WAKE_FL	QSPI Wakeup Flags Register
0x002C	RW	QSPI_WAKE_EN	QSPI Wakeup Enable Register
0x0030	R	QSPI_STAT	QSPI Status Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

15.5.1 QSPI FIFO Data Register (QSPI_DATA, Offset 0x0000)

Table 15-5. QSPI_DATA (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	DATA[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	DATA [23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	DATA[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	DATA[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DATA	31:0	SPI Data This register is used for the QSPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in either 1-, 2- or 4-byte widths only.	

15.5.2 QSPI Master Control Register (QSPI_MSTR_CNTL, Offset 0x0004)

Table 15-6. QSPI_MSTR_CNTL (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[4:0]					SS[2:0]		
Reset	0000 0					000		
Access	R					RW		
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							SSCTRL
Reset	0000 0000							0
Access	R							RW
Position	7	6	5	4	3	2	1	0
Field	RFU[1:0]		START	SSIO	RFU[1:0]		MMEN	SPIEN
Reset	00		0	0	0		0	0
Access	R		R1W	RW	RW		RW	RW

Name	Bits	Description	Settings
SPIEN	0	SPI Enable. Disabling does not affect the QSPI FIFO or register settings.	0: SPI operation is disabled. 1: SPI operation is enabled.
MMEN	1	SPI Master Mode Enable.	0: SPI configured in slave mode. 1: SPI configured in master mode.
RFU	3:2	Reserved	
SSIO	4	Master Slave Select Signal Direction. This bit is only used when the QSPI is configured for Master Mode	0: Slave Select is an output 1: Slave Select is an input
START	5	Master Start Data Transmission. Set this field to 1 to start the transaction. This bit is only used when the QSPI is configured for Master Mode	0: No Master Mode transaction active 1: Master initiates a data transmission. Ensure that all pending transactions are complete before setting this field to 1.
RFU	7:6	Reserved	
SSCTRL	8	Master Slave Select Control. This field controls the behavior of the slave select pins at the completion of a transaction. The default behavior, ssctrl = 0, deasserts the slave select pin at the completion of the transaction. Set this field to 1 to leave the slave select pins asserted at the completion of the transaction. If the external device supports this behavior, leaving the slave select pins asserted allows multiple transactions without the delay associated with deassertion of the slave select pin between transactions	0: Slave Select is deassertion of the end of a transmission 1: Slave Select stays asserted at the end of a transmission.
RFU	15:9	Reserved	

Name	Bits	Description	Settings
SS	18:16	Master Slave Select. The QSPI includes up to three slave select lines for each port. This field selects which slave select pin is active when the next SPI transaction is started. One or more slave select pins can be selected for each SPI transaction by setting the bit for each slave select pin. For example, use QSPI_SS0 and QSPIn_SS2 by setting this field to 0b101. Note: This field is only used when the QSPIn is configured for Master Mode	0bxx1: QSPI_SS0 enabled 0bx1x: QSPI_SS1 enabled 0b1xx: QSPI_SS2 enabled
RFU	31:19	Reserved	

15.5.3 SPI Transmit Packet Size Register (QSPI_TRNMT_SIZE, Offset 0x0008)

Table 15-7. QSPI_TRNMT_SIZE (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RX_NUM_CHAR[15:8]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	RX_NUM_CHAR[7:0]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	TX_NUM_CHAR[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	TX_NUM_CHAR[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
TX_NUM_CHAR	15:0	Number of Transmit Characters. Number of characters to transmit from TX FIFO. Note: If the QSPI port is set to operate in 4-wire mode, this field is used for both the number of characters to receive and transmit.	
RX_NUM_CHAR	31:16	Number of Receive Characters. Number of characters to receive in RX FIFO. Note: If the QSPI port is set to operate in 4-wire mode, this field is ignored and the TX_NUM_CHAR field is used for both the number of characters to receive and transmit.	

15.5.4 SPI Static Configuration (QSPI_STATIC_CONFIG, Offset 0x000C)

Table 15-8. SPI_STATIC_CONFIG (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[3:0]				SSPOL			
Reset	0000				0000			
Access	R				RW			
Position	15	14	13	12	11	10	9	8
Field	3WIRE	RFU[4:0]	DATAWIDTH[1:0]		NUMBITS[3:0]			
Reset	0	0000 0	00		0000			
Access	RW	R	RW		RW			
Position	7	6	5	4	3	2	1	0
Field	RFU						CLKPOL	PHASE
Reset	0000 00						0	0
Access	R						RW	RW

Name	Bits	Description	Settings
PHASE	0	Clock Phase.	0: Data sampled on clock rising edge. Use when in SPI Mode 0 and Mode 2 1: Data sampled on clock falling edge. Use when in SPI Mode 1 and Mode 3
CLKPOL	1	Clock Polarity. This field controls the SCK polarity. The default clock polarity is for SPI Mode 0 and Mode 1 operation and is active high. Invert the SCK polarity for SPI Mode 2 and Mode 3 operation.	0: Standard SCK for use in SPI Mode 0 and Mode 1 1: Inverted SCK for use in SPI Mode 2 and Mode 3
RFU	7:2	Reserved	
NUMBITS[3:0]	11:8	Number of Data Bits per Character to Transfer	This field contains the number of bits to shift for each character transfer. Refer to the data movement chapter for information on valid bit positions when the character length is less than 16-bits. NUMBITS Number of bits per character 0000 16 0001 1 0010 2 1110 14 1111 15 Note: 1-bit and 9-bit character lengths are not supported in Slave Mode, QSPI_MSTR_CNTL.MMEN = 0. Note: For Dual and Quad mode SPI, the character size should be divisible by the number of bits per SCK cycle.
DATAWIDTH	13:12	SPI Data Width This field controls the number of data lines used for SPI communications. Three-wire SPI, datawidth = 0 Set this field to 0, indicating QSPI_MOSI (SISO) is used for half duplex communication. Four-wire full-duplex SPI operation, datawidth = 0 Set this field to 0, indicating QSPI_MOSI and QSPI_MISO are used for the SPI data output and input respectively. Dual SPI, datawidth = 1, uses QSPI_SDIO0 and QSPI_SDIO1 for half-duplex communication. Quad SPI, datawidth = 2, uses QSPI_SDIO0, QSPI_SDIO1, QSPI_SDIO2 and QSPI_SDIO3 for half-duplex communication.	0: 1-bit per SCK cycle (Three-wire half duplex SPI and Four-wire full duplex SPI) 1: 2-bits per SCK cycle (Dual SPI) 2: 4-bits per SCK cycle (Quad SPI) 3: Reserved Note: When this field is set to 0, use the field QSPI_STATIC_CONFIG.3WIRE to select either Three-Wire SPI or Four-Wire SPI operation.
RFU	14	Reserved	
3WIRE	15	Three-Wire SPI Enable Set this field to 1 to enable three-wire SPI communication. Set this field to 0 for four-wire full-duplex SPI communication.	0: Four-wire full-duplex SPI mode enabled 1: Three-wire mode enabled Note: This field is ignored for Dual SPI, Quad SPI.
SSPOL	19:16	Slave Select Polarity. Controls the polarity of each individual SS signal where each bit position corresponds to a SS signal. QSPI_SS0 is controlled with bit position 0 and QSPI_SS2 is controlled with bit position 2.	For each bit position, 0: SS is active low 1: SS is active high
RFU	31:20	Reserved.	

15.5.5 QSPI Slave Select Timing Register (QSPI_SS_TIME, Offset 0x0010)

Table 15-9. QSPI_SS_TIME (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	INACT[7:0]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	POST[7:0]							
Reset	0000 0000							
Access	R				RW			
Position	7	6	5	4	3	2	1	0
Field	PRE[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
PRE	7:0	Slave Select Delay to First SCK Set the number of system clock cycles the Slave Select is held active prior to the first SCK edge.	0: 256 1: 1 2: 2 3:3 254: 254 255: 255
POST	15:8	Slave Select Hold Post Last SCK Number of system clock cycles that SS remains active after the last SCK edge.	0: 256 1: 1 2: 2 3:3 254: 254 255: 255
INACT	23:16	Inactive Stretch This field controls the number of system clocks the bus is inactive between the end of a transaction (Slave Select inactive) and the start of the next transaction (Slave Select active).	0: 256 1: 1 2: 2 3:3 254: 254 255: 255
RFU	31:24	Reserved	

15.5.6 QSPI Master Clock Configuration Register (QSPI_CLK_CONFIG, Offset 0x0014)

Table 15-10. QSPI_CLK_CONFIG (Offset 0x0014)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[3:0]				SCALE[3:0]			
Reset	0000				0000			
Access	R				RW			
Position	15	14	13	12	11	10	9	8
Field	HIGH[7:0]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	LOW[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
LOW	7:0	SCK Low Clock Cycles Control. This field controls the SCK low clock time and is used to control the overall SCK duty cycle in combination with the QSPI_CLK_CFG.HIGH field.	0: Low duty cycle control disabled. Setting this field to 0 is only valid if QSPI_CLK_CFG.SCALE = 0. 1 to 15: The number of QSPI peripheral clocks, $f_{QSPICLK}$, that the SCK signal is low. Note: If QSPI_CLK_CFG.SCALE = 0, QSPI_CLK_CFG.HIGH = 0, and QSPI_CLK_CFG.LOW = 0, character sizes of 2 and 10 bits are not supported.
HIGH	15:8	SCK High Clock Cycles Control. This field controls the SCK HIGH clock time and is used to control the overall SCK duty cycle in combination with the QSPI_CLK_CFG.LOW field.	0: Hi duty cycle control disabled. Only valid if scale = 0. 1 to 15: The number of QSPI peripheral clocks, $f_{QSPICLK}$, that SCK is high. Note: If QSPI_CLK_CFG.SCALE = 0, QSPI_CLK_CFG.HIGH = 0, and QSPI_CLK_CFG.LOW = 0, character sizes of 2 and 10 bits are not supported.
SCALE	19:16	SPI Peripheral Clock Scale. Scales the QSPI input clock (PCLK for QSPI0/QSPI1 and HCLK for QSPI2) by 2^{scale} to generate the QSPI peripheral clock. $f_{QSPICLK} = \frac{f_{QSPIN\_INPUT\_CLK}}{2^{\text{scale}}}$	Valid values for scale are 0 to 8 inclusive. Values greater than 8 are reserved for future use. Note: If QSPI_CLK_CFG.SCALE = 0, QSPI_CLK_CFG.HIGH = 0, and QSPI_CLK_CFG.LOW = 0, character sizes of 2 and 10 bits are not supported.
RFU	31:20	Reserved	

15.5.7 QSPI DMA Register (QSPI_DMA, Offset 0x001C)

Table 15-11. QSPI_DMA (Offset 0x001C)

Position	31	30	29	28	27	26	25	24
Field	RX_DMA_EN	RFU	RX_FIFO_CNT[5:0]					
Reset	0	0	00 0000					
Access	RW	R	R					
Position	23	22	21	20	19	18	17	16
Field	RX_FIFO_CLEAR	RX_FIFO_EN	RFU	RX_FIFO_LEVEL[4:0]				
Reset	0	0	0	0 0000				
Access	W	RW	R	RW				
Position	15	14	13	12	11	10	9	8
Field	TX_DMA_EN	RFU	TX_FIFO_CNT[5:0]					
Reset	0	0	00 0000					
Access	RW	R	R					
Position	7	6	5	4	3	2	1	0
Field	TX_FIFO_CLEAR	TX_FIFO_EN	RFU	TX_FIFO_LEVEL[4:0]				
Reset	0	0	0	1 0000				
Access	RW	RW	R	RW				

Name	Bits	Description	Settings
TX_FIFO_LEVEL	4:0	Transmit FIFO Threshold Level. When the TX FIFO count (QSPI_DMA.TX_FIFO_CNT) falls below this value, a DMA request is triggered and QSPI_INT_FL.TX_THR is set.	
RFU	5	Reserved	
TX_FIFO_EN	6	TX FIFO Enabled. Set this field to enable the TX FIFO.	0: TX FIFO disabled 1: TX FIFO enabled
TX_FIFO_CLEAR	7	TX FIFO Clear. Set this bit to clear the TX FIFO and all TX FIFO flags in the QSPI_INT_FL register. Note: The TX FIFO should be disabled (QSPI_DMA.TX_FIFO_EN = 0) prior to setting this field. Note: Setting this field to 0 has no effect.	
TX_FIFO_CNT	13:8	Number of Bytes in the TX FIFO. Read this field to determine the number of bytes currently in the TX FIFO.	
RFU	14	Reserved	
TX_DMA_EN	15	TX DMA Enable. Set this field to 1 to enable transmit DMA.	0: TX DMA is disabled. Any pending DMA requests are cleared 1: TX DMA is enabled
RX_FIFO_LEVEL	20:16	RX FIFO Threshold Level. Set this value to the desired RX FIFO threshold level. When the RX FIFO contains the number of bytes or greater than this field, a DMA request is triggered, and QSPI_INT_FL.RX_THRESH is set.	Valid values are 0 to 30. Note: 31 is an invalid setting and reserved for future use.
RFU	21	Reserved	
RX_FIFO_EN	22	RX FIFO Enabled. Set this field to enable the RX FIFO	0: RX FIFO disabled 1: RX FIFO enabled
RX_FIFO_CLEAR	23	RX FIFO Clear Set this bit to clear the RX FIFO and all the RX FIFO flags in the QSPI_INT_FL register. Note: The RX FIFO should be disabled (QSPI_DMA.RX_FIFO_EN = 0) prior to setting this field. Note: Setting this field to 0 has no effect.	

Name	Bits	Description	Settings
RX_FIFO_CNT	29:24	Number of Bytes in the RX FIFO. Read returns the number of bytes currently in the RX FIFO	
RFU	30	Reserved	
RX_DMA_EN	31	TX DMA Enable. Set this field to 1 to enable transmit DMA.	0: TX DMA is disabled. Any pending DMA requests are cleared 1: TX DMA is enabled

15.5.8 QSPI Interrupt Status Register (QSPI_INT_FL, Offset 0x0020)

Table 15-12. QSPI_INT_FL (Offset 0x0020)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RXUNDR	RXOVR	TXUNDR	TXOVR	MSTRDONE	RFU	ABORT	FAULT
Reset	0	0	0	0	0	0	0	0
Access	R	R	RW1C	RW1C	RW1C	R	RW1C	RW1C
Position	7	6	5	4	3	2	1	0
Field	RFU[1:0]		SSD	SSA	RXFULL	RXTHRLD	TXEMPTY	TXTHRLD
Reset	00		0	0	0	0	1	0
Access	R		RW1C	RW1C	RW1C	RW1C	RW1C	RW1C

Name	Bits	Description	Settings
TXTHRLD	0	TX FIFO Threshold Level Crossed Flag. Set when the TX FIFO is less than the value in QSPI_DMA.TX_FIFO_LEVEL.	
TXEMPTY	1	TX FIFO Empty Flag	
RXTHRLD	2	RX FIFO Threshold Level Crossed Flag. Set when the RX FIFO exceeds the value in QSPI_DMA.TX_FIFO_LEVEL.	
RXFULL	3	RX FIFO Full Flag	
SSA	4	Slave Select Asserted Flag	
SSD	5	Slave Select Deasserted Flag	
RFU	7:6	Reserved	
FAULT	8	Multi-Master Fault Flag. Set if the SPI is in Slave Mode, and SS is deasserted before a complete character is received.	
ABORT	9	Slave Mode Transaction Abort Detected Flag. Set if the SPI is in Slave Mode, and SS is deasserted before a complete character is received.	
RFU	10	Reserved	
MSTRDONE	11	Master Data Transmission Done Flag. Set if SPI is in Master Mode, and all transactions have completed.	
TXOVR	12	TX FIFO Overrun Flag. Set when a write is attempted to a full TX FIFO.	
TXUNDR	13	TX FIFO Underrun Flag. Set if SPI is in Slave Mode, and a read from empty TX FIFO is attempted. If SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is written to the empty TX FIFO.	
RXOVR	14	RX FIFO Overrun Flag. Set if SPI is in Slave Mode, and a write to a full RX FIFO is attempted. If the SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is read from the RX FIFO.	

Name	Bits	Description	Settings
RXUNDR	15	RX FIFO Underrun Flag. Set when a read is attempted from an empty RX FIFO.	
RFU	31:16	Reserved	

15.5.9 QSPI Interrupt Enable Register (QSPI_INT_EN, Offset 0x0024)

Table 15-13. QSPI_INT_EN (Offset 0x0024)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RXUNDR	RXOVR	TXUNDR	TXOVR	MSTRDONE	RFU	ABORT	FAULT
Reset	0	0	0	0	0	0	0	0
Access	RW	RW	RW	RW	RW	R	RW	RW
Position	7	6	5	4	3	2	1	0
Field	RFU[1:0]		SSD	SSA	RXFULL	RXTHRLD	TXEMPTY	TXTHRLD
Reset	00		0	0	0	0	0	0
Access	R		RW	RW	RW	RW	RW	RW

Name	Bits	Description	Settings
TXTHRLD	0	TX FIFO Threshold Level Crossed Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
TXEMPTY	1	TX FIFO Empty Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
RXTHRLD	2	RX FIFO Threshold Level Crossed Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
RXFULL	3	RX FIFO Full Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
SSA	4	Slave Select Asserted Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
SSD	5	Slave Select Deasserted Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
RFU	7:6	Reserved for Future Use	
FAULT	8	Multi-Master Fault Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
ABORT	9	Slave Mode Abort Detected Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
RFU	10	Reserved	
MSTRDONE	11	Master Data Transmission Done Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
TXOVR	12	TX FIFO Overrun Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
TXUNDR	13	TX FIFO Underrun Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
RXOVR	14	RX FIFO Overrun Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled
RXUNDR	15	RX FIFO Underrun Interrupt Enable	0: Interrupt is disabled 1: Interrupt is enabled

Name	Bits	Description	Settings
RFU	31:16	Reserved for Future Use	

15.5.10 QSPI Wakeup Status Register (QSPI_WAKE_FL, Offset 0x0028)

Table 15-14. QSPI_WAKE_FL (Offset 0x0028)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[3:0]				RXFULL	RXTHRLD	TXEMPTY	TXTHRLD
Reset	0000				0	0	0	0
Access	R				RW	RW1C	RW1C	RW1C

Name	Bits	Description	Settings
TXTHRLD	0	Wake on TX FIFO Threshold Level Crossed Flag	0: Wake condition has not occurred. 1: Wake condition occurred.
TXEMPTY	1	Wake on TX FIFO Empty Flag	0: Wake condition has not occurred. 1: Wake condition occurred.
RXTHRLD	2	Wake on RX FIFO Threshold Level Crossed Flag	0: Wake condition has not occurred. 1: Wake condition occurred.
RXFULL	3	Wake on RX FIFO Full Flag	0: Wake condition has not occurred. 1: Wake condition occurred.
RFU	31:4	Reserved for Future Use	

15.5.11 QSPI Wakeup Enable Register (QSPI_WAKE_EN, Offset 0x002C)

Table 15-15. QSPI_WAKE_EN (Offset 0x002C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[3:0]				RXFULL	RXTHRLD	TXEMPTY	TXTHRLD
Reset	0000				0	0	0	0
Access	R				RW	RW1C	RW1C	RW1C

Name	Bits	Description	Settings
TXTHRLD	0	Wake on TX FIFO Threshold Level Crossed Enable	0: Wake event is disabled. 1: Wake event is enabled.
TXEMPTY	1	Wake on TX FIFO Empty Enable	0: Wake event is disabled. 1: Wake event is enabled.
RXTHRLD	2	Wake on RX FIFO Threshold Level Crossed Enable	0: Wake event is disabled. 1: Wake event is enabled.
RXFULL	3	Wake on RX FIFO Full Enable	0: Wake event is disabled. 1: Wake event is enabled.
RFU	31:4	Reserved for Future Use	

15.5.12 QSPI Status Register (QSPI_STAT, Offset 0x0030)

Table 15-16. QSPI_STAT (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							BUSY
Reset	0000 000							0
Access	R							R

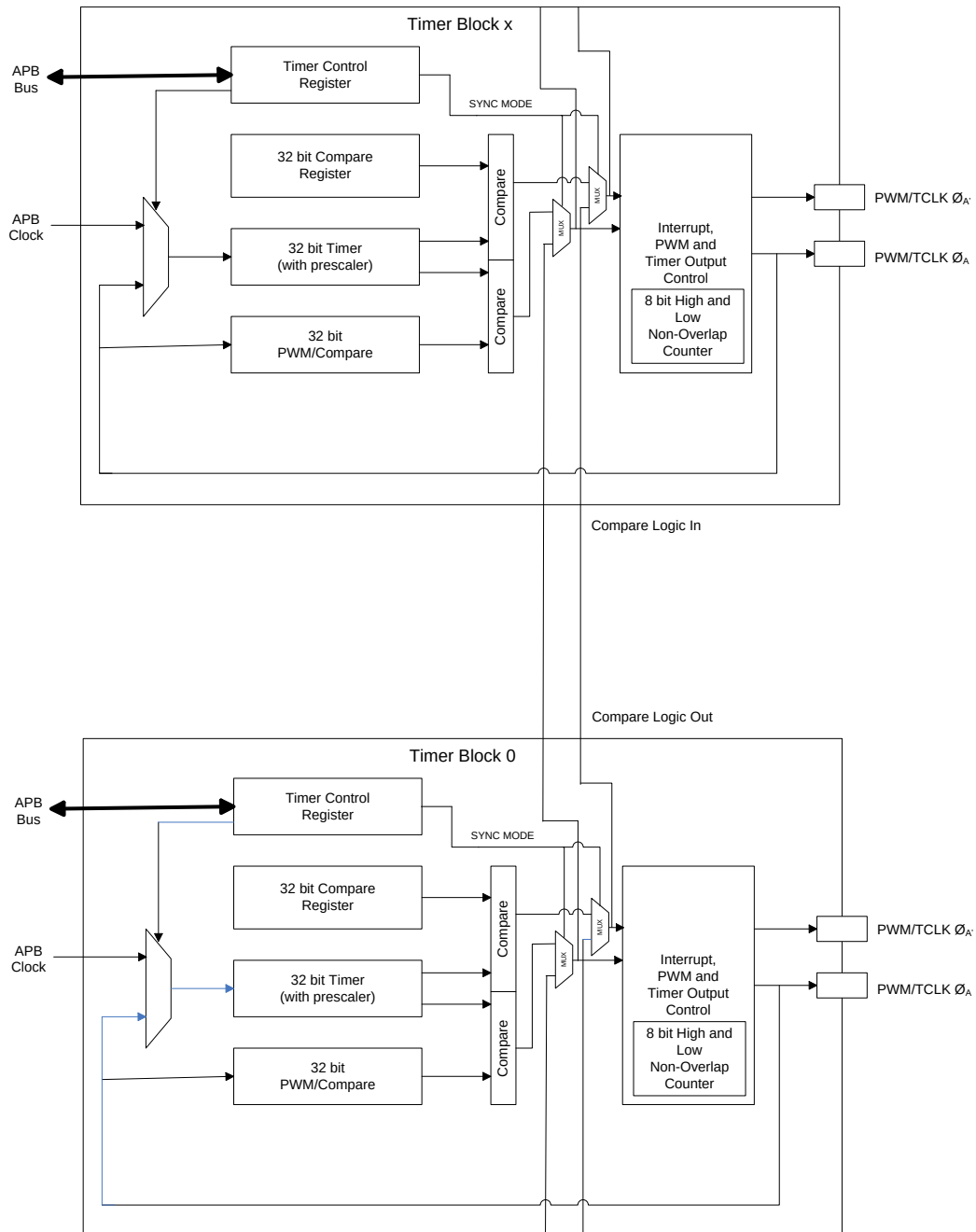
Name	Bits	Description	Settings
BUSY	0	SPI Active Status	0: QSPI is not active. In master mode, the busy flag is cleared when the last character is sent. In slave mode, the busy field is cleared when the configured slave select input is deasserted. 1: QSPI is active. In master mode, the busy flag is set when a transaction starts. In slave mode, the busy flag is set when a configured slave select input is asserted.
RFU	31:1	Reserved for Future Use	

16.0 Timers

There are four 32-bit reloadable timers that can be used for timing and event counting, four of which may be used for the generation of pulse-width modulated (PWM) signals. The timers' features include:

- 32-bit reload counter
- Programmable prescaler with prescale values from 1 to 4096
- Non-overlapping Hi-Drive (8mA) PWM output generation with configurable off-time
- Capture, compare and capture/compare capability
- External input pin for timer input, clock gating, or capture signal
- Timer output pin
- Timer interrupt

Figure 16-1. 32 bit Timer Block Diagram



16.1 Basic Operation

The timers are 32-bit up-counter timers. Minimum timeout delay is set by:

- Loading the value 0x0000_0001 into the Timer Compare register
- Setting the prescale value to 1

Maximum timeout delay is set by:

- Loading the value 0x0000_0000 into the Timer Compare register
- Setting the prescale value to 4096.

If the Timer reaches 0xFFFF_FFFF, the timer rolls over to 0x0000_0000 and continues counting.

The current count value in the timers can be read while the timer is counting (TEN = '1'). This action has no effect on the timer operation.

Generally, the timer output is toggled every time the counter is reloaded.

16.2 Timer Operating Modes

The timers can be configured to operate in one of the eight modes.

The timer operating modes include the following:

- One-shot Mode
- Continuous Mode
- Counter Mode
- PWM Mode
- Capture Mode
- Compare Mode
- Gated Mode
- Capture/Compare Mode

16.2.1 One-Shot Mode

In One-Shot mode, the timer counts from the start count value stored in the timer register up to the 32-bit compare value stored in the timer compare register. The timer input is the PCLK. Upon reaching the compare value, the timer generates an interrupt, and the count value in the timer register is reset to 0x0000_0001. Then, the timer is automatically disabled and stops counting. Also, if the timer output function is enabled in the GPIO, the timer output pin changes state for one clock cycle, then returns to polarity value (TPOL bit in the Timer Control Register). A software workaround is to set the polarity to not (TPOL) during the counting phase.

The steps for configuring a timer for one-shot mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = "0").
 - Configure the timer for one-shot mode (TMODE = "000").
 - Set the prescale value (PRES).
 - If using the Timer Output function, set the initial output level (High or Low) via TPOL.

2. Write to the Timer register to set the starting count value.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt in the Interrupt Controller and set the timer interrupt priority by writing to the appropriate Configuration Table Register in the Interrupt Controller.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. See chapter covering GPIO.
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).
7. Negate the TPOL bit during the counting phase.

In One-Shot mode, the PCLK always provides the timer input. The timer period is given by the following equation:

$$\text{Oneshot Timeout Period}(s) = \frac{\text{Reload} - \text{Start Value}}{\text{System Clock Frequency (Hz)}} \times \text{Prescale}$$

16.2.2 Continuous Mode

In Continuous mode, the timer counts up to the 32-bit Compare value stored in the Timer Compare register. The timer input is the PCLK. Upon reaching the Compare value, the timer generates an interrupt; the count value in the Timer register is reset to 0x0000_0001 and counting resumes. Also, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low to High or from High to Low upon Timer Compare.

The steps for configuring a timer for Continuous mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).
 - Configure the timer for Continuous mode (TMODE = “001”).
 - Set the prescale value (PRES).
 - If using the Timer Output function, set the initial output level (High or Low) via TPOL.
2. Write to the Timer register to set the starting count value. This only affects the first pass in Continuous mode. After the first Timer Compare in Continuous mode, counting always begins at the reset value of 0x0000_0001.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt in the Interrupt Controller and set the timer interrupt priority by writing to the appropriate Configuration Table Register in the Interrupt Controller.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. See chapter covering GPIO.
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

In Continuous mode, the PCLK always provides the timer input. The timer period is given by the following equation:

$$\text{Continuous Timeout Period (s)} = \frac{\text{Reload Value}}{\text{System Clock Frequency (Hz)}} \times \text{Prescale}$$

If an initial starting value other than 0x0000_0001 is loaded into the Timer register, the One-Shot mode equation must be used to determine the first time-out period.

16.2.3 Counter Mode

In Counter mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO Port pin Timer Input function. The TPOL bit in the Timer Control Register selects if the count occurs on the rising edge (TPOL = “0”) or the falling edge (TPOL = “1”) of the Timer Input signal. In Counter mode, the prescaler is disabled. Any value assigned to PRES in Counter Mode has no effect. The input frequency of the Timer Input signal must not exceed one-fourth of the PCLK frequency.

Upon reaching the Compare value stored in the Timer Compare register, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001 and counting resumes. Also, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low to High or from High to Low at Timer Compare.

The steps for configuring a timer for Counter mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).
 - Configure the timer for Counter mode (TMODE = “010”).
 - Select either the rising edge or falling edge of the Timer Input signal for the count (TPOL). This also sets the initial logic level (High or Low) for the Timer Output function. However, the Timer Output function does not have to be enabled (via GPIO).
2. Write to the Timer register to set the starting count value. This only affects the first pass in Counter mode. After the first Timer Compare in Counter mode, counting always begins at the reset value of 0x0000_0001. Generally, in Counter mode, the Timer register should be written with the value 0x0000_0001.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. See chapter covering GPIO.
7. Write to the Timer Control register to enable the timer (TEN = “1”).

The number of Timer Input transitions since the timer start is given by the following equation:

$$\text{Counter Mode Timer Input Transition} = \text{Current Count value} - \text{Start Value}$$

16.2.4 PWM Mode

In PWM mode, the timer outputs a Pulse-Width Modulator (PWM) output signal through a GPIO Port pin. The timer input is the PCLK. The timer first counts up to the 32-bit PWM match value stored in the Timer PWM register. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the Compare value stored in the Timer Compare register. Upon reaching the Compare value, the Timer Output signal toggles again, the timer generates an interrupt, the count value in the Timer register is reset to 0x0001 and counting resumes.

When TPOL = “0” (in TMRx_CTRL Register):

Timer Output signal begins as a Low (“0”) and then transitions to a High (“1”) when the timer value matches the PWM value. The Timer Output signal returns to a Low (“0”) after the timer reaches the Compare value, which is reset to 0x0000_0001.

When TPOL = “1” (in TMRx_CTRL Register):

Timer Output signal begins as a High (“1”) and then transitions to a Low (“0”) when the timer value matches the PWM value. The Timer Output signal returns to a High (“1”) after the timer reaches the Compare value, which is reset to 0x0000_0001.

The steps for configuring a timer for PWM mode and initiating the PWM operation are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).
 - Configure the timer for PWM mode (TMODE = “011”).
 - Set the prescale value (PRES).
 - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output function (TPOL).
2. Write to the Timer register to set the starting count value (typically 0x0000_0001). This only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of 0x0000_0001.
3. Write to the PWM register to set the Match value.
4. Write to the Timer Compare register to set the Compare value. The Compare value must be greater than the PWM value.
5. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
6. Configure the associated GPIO port pin for the Timer Output function. (See the chapter covering GPIO herein.)
7. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

The PWM period is given by the following equation:

$$PWM\ Period\ (s) = \frac{Reload\ Value}{System\ Clock\ Frequency\ (Hz)} \times Prescale$$

If an initial starting value other than 0x0000_0001 is loaded into the Timer register, the One-Shot mode equation must be used to determine the first PWM time-out period.

If TPOL is set to “0”, the ratio of the PWM output High time to the total period is given by:

$$PWM\ Output\ High\ Time\ Ratio\ (\%) = \frac{Reload\ Value - PWM\ Value}{Reload\ Value} \times 100$$

If TPOL is set to “1”, the ratio of the PWM output High time to the total period is given by:

$$PWM\ Output\ High\ Time\ Ratio\ (\%) = \frac{PWM\ Value}{Reload\ Value} \times 100$$

16.2.5 Capture Mode

In Capture mode, the current timer count value is recorded when the desired external Timer Input transition occurs. The Capture count value is written to the Timer PWM register. The timer input is the PCLK. The TPOL bit in the

Timer Control register determines if the Capture occurs on a rising edge (TPOL = “0”) or a falling edge (TPOL = “1”) of the Timer Input signal. When the Capture event occurs, an interrupt is generated, and the timer continues counting.

The timer continues counting to the 32-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes.

The steps for configuring a timer for Capture mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).
 - Configure the timer for Capture mode (TMODE = “100”).
 - Set the prescale value (PRES).
 - Set the Capture edge (rising or falling) for the Timer Input (TPOL).
2. Write to the Timer register to set the starting count value (typically 0x0000_0001).
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

In Capture mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{\text{Capture Value} - \text{Start Value}}{\text{System Clock Frequency (Hz)}} \times \text{Prescale}$$

16.2.6 Compare Mode

In Compare mode, the timer counts up to the 32-bit maximum Compare value stored in the Timer Compare register. The timer input is the PCLK. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0x0000_0001). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) upon Compare.

When the Timer reaches 0xFFFF_FFFF, the timer rolls over to 0x0000_0000 and continues counting.

The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).
 - Configure the timer for Compare mode (TMODE = “101”).
 - Set the prescale value (PRES).
 - If using the Timer Output function, set the initial logic level (High or Low) via TPOL.
2. Write to the Timer register to set the starting count value.
3. Write to the Timer Compare register to set the Compare value.

4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

In Compare mode, the PCLK always provides the timer input. The Compare time is given by the following equation:

$$\text{Compare Mode Time (s)} = \frac{\text{Compare Value} - \text{Start Value}}{\text{System Clock Frequency (Hz)}} \times \text{Prescale}$$

16.2.7 Gated Mode

In Gated mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is de-asserted or a Timer Compare occurs.

The timer counts up to the 32-bit Compare value stored in the Timer Compare register. The timer input is the PCLK. When reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001 and counting resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) at Timer Compare.

The steps for configuring a timer for Gated mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).
 - Configure the timer for Gated mode (TMODE = “110”).
 - Set the prescale value (PRES).
2. Write to the Timer register to set the starting count value. This only affects the first pass in Gated mode. After the first timer reset in Gated mode, counting always begins at the reset value of 0x0000_0001.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)
7. Write to the Timer Control register to enable the timer (TEN = “1”).
8. Assert the Timer Input signal to initiate the counting (High if TPOL = “0”; Low if TPOL = “1”).

16.2.8 Capture/Compare Mode

In Capture/Compare mode, the timer begins counting after the first desired external Timer Input transition occurs. The desired transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control Register. The timer input is the PCLK.

Every subsequent desired transition (after the first) of the Timer Input signal captures the current count value. The Captured timer value is written to the Timer PWM register. When the Capture event occurs, an interrupt is generated, the count value in the Timer register is reset to 0x0001, and counting resumes.

If no Capture event occurs, the timer counts up to the 32-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0x0000_0001, and counting resumes.

The steps for configuring a timer for Capture/Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
 - Disable the timer (TEN = “0”).

- Configure the timer for Capture/Compare mode (TMODE = “111”).
 - Set the prescale value (PRES).
 - Set the Capture edge (TPOL = “0” for rising; TPOL = “1” for falling) for the Timer Input.
2. Write to the Timer register to set the starting count value (typically 0x0001).
 3. Write to the Timer Compare register to set the Compare value.
 4. If desired, enable the timer interrupt (through the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
 5. Configure the associated GPIO port pin for the Timer Input function.
 6. Write to the Timer Control register to enable the timer (TEN = “1”).
 7. Counting begins after the first desired transition of the Timer Input signal. No interrupt is generated by this first edge.

In Capture/Compare mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{\text{Capture Value} - \text{Start Value}}{\text{System Clock Frequency (Hz)}} \times \text{Prescale}$$

16.3 Timer Registers

Address assignments for Timer registers within each block are outlined in Table 2-2. Peripheral Bus Region. Reserved register bits should only be written as 0.

Each Timer instance is controlled by a block of registers assigned to that port; all blocks contain identical registers for control, interrupt etc. Address assignments for the Timer registers are outlined in Table 16-1. Timer Register Address Offset (for Each Instance). Reserved register bits should only be written as 0.

Register names for different instances can be defined by appending the instance number to the peripheral name. So, for instance, the Timer Count Register for Timer 0 would be TMR0_CNT, while the Timer Count Register for Timer 1 would be TMR1_CNT, and so on.

Table 16-1. Timer Register Address Offset (for Each Instance)

Offset	Access	Register	Description
0x0000	RW	TMR_CNT	Timer Count Register
0x0004	RW	TMR_CMP	Timer Compare Register
0x0008	RW	TMR_PWM	Timer PWM Register
0x000C	RW	TMR_INT	Timer Interrupt Register
0x0010	RW	TMR_CN	Timer Control Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

16.3.1 Timer Count Register (TMR_CNT, Offset 0x0000)

Table 16-2. TMR_CNT (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	CNT[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	CNT[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	CNT[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	CNT[7:0]							
Reset	0000 0001							
Access	RW							

Name	Bits	Description	Settings
CNT	31:0	Count. This register stores the current timer count.	

16.3.2 Timer Compare Register (TMR_CMP, Offset 0x0004)

Table 16-3. TMR_CMP (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	CMP[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	CMP[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	CMP[15:8]							
Reset	1111 1111							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	CMP[7:0]							
Reset	1111 1111							
Access	RW							

Name	Bits	Description	Settings
CMP	31:0	Compare. This register stores the compare value, which is used to set the maximum count value to initiate a reload of the timer to 0x0001.	

16.3.3 Timer PWM Register (TMR_PWM, Offset = 0x0008)

Table 16-4. TMR_PWM (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	PWM[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	PWM[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	PWM[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	PWM[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
PWM	31:0	PWM. This register stores the value that is compared to the current timer count.	

16.3.4 Timer Interrupt Register (TMR_INT, Offset 0x000C)

Table 16-5. TMR_INT (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[6:0]							IRQ_CLR
Reset	0000 000							0
Access	R							RW

Name	Bits	Description	Settings
IRQ_CLR	0	Clear Interrupt. Writing a value (0 or 1) to a bit in this register clears the associated interrupt.	
RFU	31:1	Reserved	N/A

16.3.5 Timer Control Register (TMR_CN, Offset 0x0010)

Table 16-6. TMR_CN (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[2:0]			PWMCKBD	NOLLPOL	NOLHPOL	PWMSYNC	PRES.3
Reset	000			0	0	0	0	0
Access	R			RW	RW	RW	RW	RW
Position	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES[2:0]			TMODE[2:0]		
Reset	0	0	000			000		
Access	RW	RW	RW			RW		

Name	Bits	Description	Settings
TMODE	2:0	Timer Mode.	000: One-Shot mode 001: Continuous mode 010: Counter mode 011: PWM mode 100: Capture mode 101: Compare mode 110: Gated mode 111: Capture/Compare mode
PRES	5:3	Prescaler. The prescaler bits are formed from PRES.3:PRES[2:0]	0000: /1 0001: /2 0010: /4 0011: /8 0100: /16 0101: /32 0110: /64 0111: /128 1000: /256 1001: /512 1010: /1024 1011: /2048 1100: /4096 1101: undefined
TPOL	6	Timer input/output polarity bit, refer to 16.2.4, PWM Mode section for details.	
TEN	7	Timer Enable.	0: Timer disabled. 1: Timer enabled.
PRES.3	8	MSB of prescaler value.	
PWMSYNC	9	This bit must be set to 0.	
NOLHPOL	10	This bit must be set to 0.	
NOLLPOL	11	This bit must be set to 0.	
PWMCKBD	12	This bit must be set to 1.	
RFU	31:13	Reserved	N/A

17.0 True Random Number Generator

17.1 Overview

A True Random Number Generator (TRNG) is a critical component in most cryptographic applications. Random numbers are often used as confidential keys and initialization vectors.

The TRNG has the following features:

- NIST SP 800-90B compliant including
 - Optional conditioning based on LFSR
- 32-bit Random data register
- NIST complaint health tests
 - Startup health tests
 - Continuous health tests
 - On-demand health tests
- Interrupt generation

17.2 Reading the Random Number

The TRNG block is optimized to generate a 32-bit random number. When the RNG_ST.RNDRDY bit is 1, 32-bit random data is ready to be read from TRNG_DATA register. An interrupt is generated when RNDRDY=1 if TRNG_CN.RNG_IE=1.

17.3 AES Key Generation

By setting the AESKG bit in the TRNG Control Register, a 256-bits random number is automatically generated and sent to the backed-up memory.

During the entire process (256-bits random number generation and AES key transfer through the bus), reading the TRNG Data Register returns 0x0000 0000 to prevent the AES key monitoring through the TRNG Data Register. The entire process is transparent from the user and software standpoint.

By setting TRNG_CN.AESKG_ENC a key for memory encryption is generated and transferred to the secure key register automatically without user visibility or intervention.

17.4 Health Tests

Health tests are an integral part of the entropy source design to ensure the noise source of the entire entropy continue to operate as expected. The TRNG block includes the following tests.

17.4.1 Startup tests

During the start-up, continuous tests runs on at least 4096 consecutive samples from entropy source. If the test fails, TRNG_ST.HLTH_ST bit is set.

17.4.2 Repetition count test

The goal of this test to quickly detect catastrophic failures that cause the noise source to 'stuck' on zero or one.

17.4.3 Adaptive proportion test

According to NIST SP-800-90B, the Adaptive Proportion Test is designed to detect a large loss of entropy that might occur because of some physical failure or environmental change affecting the noise source. The test continuously measures the local frequency of occurrence of a sample value in a sequence of noise source samples to determine if the sample occurs too frequently. Thus, the test can detect when some value begins to occur much more frequently than expected, given the source's assessed entropy per sample. The adaptive proportion test takes a sample of the random number and counts the number of the same value with a 1024 window.

17.5 Interrupt Management

When a new random number is generated, TRNG_CN.RNDRDY is set and an interrupt is fired if TRNG_CN.RNG_IE is set.

17.6 TRNG Registers

Address assignments for the TRNG are outlined below.

Table 17-1. TRNG Register Address (Base ADDR = 0x4004 D000)

Offset	Access	Register	Description
0x0000	RW	TRNG_CN	TRNG Control Register
0x0004	RW	TRNG_ST	TRNG Status Register
0x0008	R	TRNG_DATA	TRNG Data Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

17.6.1 TRNG Control Register (TRNG_CN, Offset 0x0000)

Table 17-2. TRNG_CN (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[1:0]		RFU	RFU	AESKG_MDU_ENC	HLTH_IE	RNG_IE	ODHT
Reset	00		0	0	0	0	0	0
Access	R		R	R	RW	RW	RW	RW

Name	Bits	Description	Settings
ODHT	0	On-Demand Health Test.	0: No effect 1: Start the on-demand test
RNG_IE	1	Random Number Interrupt Enable. This bit enables an interrupt to be generated when a new, 32 bit, random number is ready to be read.	0: Interrupt disabled 1: Interrupt enabled
HLTH_IE	2	Health Test Interrupt Enable. This bit enables an interrupt if one of the health tests fails.	0: Interrupt disabled 1: Interrupt enabled
AESKG_ENC	3	AES Key Generate. When set, the key is generated and transferred to the Secure Key Storage Register automatically without user visibility or intervention. This bit is cleared by the hardware once the key is transferred to the secure key register.	0: No effect. 1: Initiate AES key generation and transfer to Secure Key Storage Register.
RFU	31:4	Reserved	N/A

Note 1: Transfer must be completed by checking the TRNG_ST.AESKG_TRNSFR_ST bit before starting a new transfer or the operation is not considered by the hardware.

17.6.2 TRNG Status Register (TRNG_ST, Offset 0x0004)

Table 17-3. TRNG_ST (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU[2:0]			AESKG_TRNSFR_ST	SRCFAIL	HLTH_ST	ODHTS	RNDRDY
Reset	000			0	0	0	0	0
Access	R			R	RW1C	RW1C	R	R

Name	Bits	Description	Settings
RNDRDY	0	32-bit random data is ready to be read from TRNG_DATA register. Reading TRNG_DATA when RNDRDY=0 returns all 0s. An interrupt is generated when RNDRDY=1 if TRNG_CN.RNG_IE=1. RNDRDY is automatically cleared when the 32-bit random data is read from the TRNG_DATA register.	0: TRNG is busy 1: 32-bit random data is ready
ODHTS	1	On-demand health test status. This bit is reset once the health tests are finished.	0: On-demand health test done 1: On-demand health test ongoing
HLTH_ST	2	Health Test Status. This bit should be checked when the on-demand health test is completed (ODHTS = 0). This bit is also set when continuous or adaptive health tests report an error. An interrupt is generated if the HLTH_IE is set. Write 1 to clear this bit.	0: No error/ Passed on-demand tests 1: A self-health test has failed, or the on-demand health test has failed.
SRCFAIL	3	Entropy source has failed. An interrupt is generated if the HLTH_IE is set. Write 1 to clear this bit. Note: This bit sets after POR or when waking up from stop mode. The software must reset and double check this bit after a POR or wakeup event.	0: Entropy source works properly 1: Entropy source has failed.
AESKG_TRNSFR_ST	4	AES key transfer status.	0: AES key has not been transferred 1: AES key has been transferred
RFU	31:5	Reserved	N/A

17.6.3 TRNG Data Register (TRNG_DATA, Offset 0x0008)

Table 17-4. TRNG_DATA (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RNDDATA[31:24]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RNDDATA[23:16]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RNDDATA[15:8]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RNDDATA[7:0]							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
RNDDATA	31:0	Random Number Register Bits. This register contains a data word generated by the random number generator when the RNG_ST.RNDRDY. bit is 1. The value of this register is 0x00000000 when TRNG_ST.RNDRDY bit is 0.	

18.0 UART (Serial Port)

There is one UART provided which can be used to communicate with external devices requiring an asynchronous serial protocol.

The features of the UARTs on the device are:

- Flexible baud rate generation based on the peripheral clock nominal frequency (typically half of the system clock frequency)
- DMA capable
- Programmable word size (5 bits to 8 bits), stop bits, and parity settings
- Automatic parity and framing error detection
- 8-byte FIFO in both directions (separate read/RX and write/TX FIFOs)
- Interrupts available for frame error, parity error, RX FIFO overrun, and FIFO full/partially full conditions

18.1 Detailed Description

The following sections provide more details on configuring the UART operating modes.

18.1.1 Selecting the Format and Baud Rate

The baud rate for the UART is based on the frequency of 120MHz Oscillator (or 7.3728MHz). If the peripheral clock is divided down under system control, this does not change the baud rate frequency basis used by the UART. To set the baud rate value, write the appropriate divider values to the registers UART_BAUD0 and UART_BAUD1.

Note that the 120MHz (or 7.3728MHz) output must be selected as the system clock source for proper UART operation.

The transfer size (5 bits to 8 bits long), number of stop bits used, and parity calculation mode are all set by writing to the appropriate bits and bit fields in the UART_CTRL register.

18.1.2 Transferring and Receiving Data

The data to transfer must be placed in the TX FIFO by writing to the UART_DATA register (either directly or by using a DMA). This data is transferred out by the hardware automatically one character at a time, in the order that it was received. The status flags and associated UART interrupts can be used to monitor the FIFO status and determine when the transfer cycle or cycles have completed.

As data is received, it is loaded into the RX FIFO, and it can then be unloaded by reading from the UART_DATA register. If 8 characters are received and are not read by the CPU or DMA controller, and another character arrives on RX, an overrun error occurs and the new character is lost.

18.1.3 UART Interrupts

Interrupts can be generated by each UART instance for one or more of the following conditions.

- A framing error occurs on a received character (RX transitions out of sync to the baud clock, which is synchronized to the initial RX falling edge).
- The received character has an invalid parity bit according to the chosen parity settings.
- The number of bytes in the FIFO RX buffer reaches a configurable threshold level.
- An overrun error occurs on the FIFO RX buffer.
- The FIFO TX buffer reaches a half-empty level.
- The FIFO TX buffer has only one byte remaining.

18.1.4 DMA Transfers

The UART supports DMA transfers in both the transmit and receive directions; separate DMA channels can be connected to the RX and TX FIFO buffers. UART interrupts are not rerouted to the DMA controller and must still be handled by the CPU in the usual manner.

18.2 UART Registers

Address assignments for UART registers within each block are outlined below.

Table 18-1. UART Register Addresses (0x40020000)

Offset	Access	Register	Description
0x0000	RW	UART_CTRL	UART Control Register
0x0004	R	UART_STAT	UART Status Register
0x0008	RW	UART_INT_EN	UART Interrupt Enable Register
0x000C	RC	UART_INT_STAT	UART Interrupt Status Register
0x0010	RW	UART_BAUD0	UART Baud Rate Register 0
0x0014	RW	UART_BAUD1	UART Baud Rate Register 1
0x001C	RW	RFU	Reserved
0x0020	RW	UART_DATA	UART Data Buffer Register
0x0030	RW	UART_DMA	UART DMA Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

18.2.1 UART Control Register (UART_CTRL, Offset 0x0000)

Table 18-2. UART_CTRL (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU			STOP	SIZE1	SIZE0	RXFLUSH	TXFLUSH
Reset	000			0	0	0	0	0
Access	R			RW	RW	RW	RW	RW
Position	7	6	5	4	3	2	1	0
Field	RFU	PARMD	PAREO	PAREN	RXTHD[3:0]			
Reset	0	0	0	0	0000			
Access	R	RW	RW	RW	RW			

Name	Bits	Description	Settings
RXTHD[3:0]	3:0	Number of bytes that must be loaded into the receive FIFO before the FFRXIS interrupt is triggered.	0: Ignored 1-8: Interrupt is triggered when the number of bytes in the RX FIFO reaches this level or higher 9+: Ignored
PAREN	4	Enables/disables generation and testing of parity bit.	0: Parity is disabled. 1: Parity is enabled.
PAREO	5	Selects odd or even parity (when PAREN=1).	0: Even parity selected. 1: Odd parity selected.
PARMD	6	Selects parity based on 1s or 0s count (when PAREN=1).	0: Parity calculation is based on number of 1s in frame. 1: Parity calculation is based on number of 0s in frame.
RFU	7	Reserved.	N/A
TXFLUSH	8	Flushes the TX FIFO buffer.	Write to 1 to flush the FIFO.
RXFLUSH	9	Flushes the RX FIFO buffer.	Write to 1 to flush the FIFO.
SIZE[1:0]	11:10	Selects UART character size.	00: 5 bits 01: 6 bits 10: 7 bits 11: 8 bits
STOP	12	Selects the number of stop bits that are generated.	0: 1 stop bit 1: 1.5 stop bits (for 5-bit mode) or 2 stop bits (for 6/7/8 bit mode)
RFU	31:13	Reserved.	N/A

18.2.2 UART Status Register (UART_STAT, Offset 0x0004)

Table 18-3. UART_STAT (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	TXELT				RXELT			
Reset	0000				0000			
Access	R				R			
Position	7	6	5	4	3	2	1	0
Field	TXFULL	TXEMPTY	RXFULL	RXEMPTY	RFU	RFU	RXBUSY	TXBUSY
Reset	0	1	0	1	0	0	0	0
Access	R	R	R	R	R	R	R	R

Name	Bits	Description	Settings
TXBUSY	0	Read-only flag indicating the UART transmit status.	0: UART is not currently transmitting data. 1: UART is currently transmitting data.
RXBUSY	1	Read-only flag indicating the UART receiver status.	0: UART is not currently receiving a character. 1: UART is currently receiving a character.
RFU	3:2	Reserved.	N/A
RXEMPTY	4	Read-only flag indicating the RX FIFO state.	0: RX FIFO is not empty. 1: RX FIFO is empty (RXELT=0).
RXFULL	5	Read-only flag indicating the RX FIFO state.	0: RX FIFO is not full. 1: RX FIFO is full (RXELT=8).
TXEMPTY	6	Read-only flag indicating the TX FIFO state.	0: TX FIFO is not empty. 1: TX FIFO is empty (TXELT=0).
TXFULL	7	Read-only flag indicating the TX FIFO state.	0: TX FIFO is not full. 1: TX FIFO is full (TXELT=8).
RXELT	11:8	Read-only field.	Indicates the number of bytes currently in the RX FIFO (0-8)
TXELT	15:12	Read-only field.	Indicates the number of bytes currently in the TX FIFO (0-8)
RFU	31:16	Reserved.	N/A

18.2.3 UART Interrupt Enable Register (UART_INT_EN, Offset 0x0008)

Table 18-4. UART_INT_EN (Offset 0x0008)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU	FFTXHIE	FFTXOIE	FFRXIE	OVERIE	RFU	PARITYIE	FRAME
Reset	0	0	0	0	0	0	0	0
Access	R	RW	RW	RW	RW	R	RW	RW

Name	Bits	Description	Settings
FRAME	0	Enable for RX Frame Error Interrupt.	0: Interrupt disabled. 1: Interrupt enabled.
PARITYIE	1	Enable for RX Parity Error interrupt.	0: Interrupt disabled. 1: Interrupt enabled.
RFU	2	Reserved	N/A
OVERIE	3	Enable for RX FIFO Overrun interrupt.	0: Interrupt disabled. 1: Interrupt enabled.
FFRXIE	4	Enable for interrupt when RX FIFO reaches the number of bytes configured by the RXTHD field.	0: Interrupt disabled. 1: Interrupt enabled.
FFTXOIE	5	Enable for interrupt when TX FIFO has only one byte remaining.	0: Interrupt disabled. 1: Interrupt enabled.
FFTXHIE	6	Enable for interrupt when TX FIFO is half empty or less.	0: Interrupt disabled. 1: Interrupt enabled.
RFU	31:7	Reserved.	N/A

18.2.4 UART Interrupt Status Register (UART_INT_STAT, Offset 0x000C)

Table 18-5. UART_INT_STAT (Offset 0x000C)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU	FFTXHIS	FFTXOIS	FFRXIS	OVERIS	RFU	PARITYIS	FRAMIS
Reset	0	0	0	0	0	0	0	0
Access	R	RC	RC	RC	RC	R	RC	RC

Name	Bits	Description	Settings
FRAMIS	0	Flag for RX Frame Error Interrupt.	Set to 1 by hardware when interrupt occurs; software must write to 0 to clear. Writes to 1 by software are ignored.
PARITYIS	1	Flag for RX Parity Error interrupt.	Set to 1 by hardware when interrupt occurs; software must write to 0 to clear. Writes to 1 by software are ignored.
RFU	2	Reserved	N/A
OVERIS	3	Flag for RX FIFO Overrun interrupt.	Set to 1 by hardware when interrupt occurs; software must write to 0 to clear. Writes to 1 by software are ignored.
FFRXIS	4	Flag for interrupt when RX FIFO reaches the number of bytes configured by the RXTHD field.	Set to 1 by hardware when interrupt occurs; software must write to 0 to clear. Writes to 1 by software are ignored.
FFTXOIS	5	Flag for interrupt when TX FIFO has only one byte remaining.	Set to 1 by hardware when interrupt occurs; software must write to 0 to clear. Writes to 1 by software are ignored.
FFTXHIS	6	Flag for interrupt when TX FIFO is half empty or less.	Set to 1 by hardware when interrupt occurs; software must write to 0 to clear. Writes to 1 by software are ignored.
RFU	31:7	Reserved.	N/A

18.2.5 UART Baud Rate Register 0 (UART_BAUD0, Offset 0x0010)

Table 18-6. UART_BAUD0 (Offset 0x0010)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU				IDIV[11:8]			
Reset	0000				0000			
Access	R				RW			
Position	7	6	5	4	3	2	1	0
Field	IDIV[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
IDIV[11:0]	11:0	Integer portion of baud rate divisor value.	Use the following formula to obtain the proper value for this field: $IDIV = (\text{integer portion of}) \text{ 'Nominal Frequency' } / (128 \times \text{Baud Rate Frequency})$ where the 'Nominal Frequency' = $2 \times f_{\text{SYSTEM CLOCK}}$ The integer portion is rounded value (i.e., 5.7 becomes 5)
RFU	31:12	Reserved.	N/A

18.2.6 UART Baud Rate Register 1 (UART_BAUD1, Offset 0x0014)

Table 18-7. UART_BAUD1 (Offset 0x0014)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	RFU	DDIV[6:0]						
Reset	0	00 0000						
Access	R	RW						

Name	Bits	Description	Settings
DDIV	6:0	Decimal portion of baud rate divisor value.	Use the following formula to obtain the proper value for this field after obtaining IDIV: $DIV = \text{'Nominal Frequency'} / (128 \times \text{Baud Rate Frequency})$ $DDIV = (DIV - IDIV) \times 128$ where the 'Nominal Frequency' = $2 \times f_{\text{SYSTEM CLOCK}}$
RFU	31:7	Reserved.	N/A

18.2.7 UART Data Buffer Register (UART_DATA, Offset 0x0020)

Table 18-8. UART_DATA (Offset 0x0020)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[6:0]							PARITY
Reset	000 0000							0
Access	R							R
Position	7	6	5	4	3	2	1	0
Field	DATA[7:0]							
Reset	0000 0000							
Access	RW							

Name	Bits	Description	Settings
DATA[7:0]	7:0	Load/unload location for TX and RX FIFO buffers.	Writing to this field loads the next character into the TX FIFO (if space is available). Reading from this field returns the next character available at the output of the RX FIFO (if one is available, otherwise 00h is returned).
PARITY	8	Parity error flag for next byte to be read from FIFO.	If a parity error occurred during the reception of the character at the output end of the RX FIFO (that is returned by reading the DATA field), this bit reads 1, otherwise it reads 0.
RFU	31:9	Reserved.	N/A

18.2.8 UART DMA Register (UART_DMA, Offset 0x0030)

Table 18-9. UART_DMA (Offset 0x0030)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[5:0]						RXEN	RXCNT3
Reset	00 0000						0	0
Access	R						RW	RW
Position	7	6	5	4	3	2	1	0
Field	RXCNT[2:0]			TXEN	TXCNT[3:0]			
Reset	000			0	0000			
Access	RW			RW	RW			

Name	Bits	Description	Settings
TXCNT	3:0	TX threshold for DMA transmission	
TXEN	4	TX DMA channel enable	0: Disabled 1: Enabled
RXCNT	8:5	RX threshold for DMA transmission	
RXEN	9	RX DMA channel enable	0: Disabled 1: Enabled
RFU	31:10	Reserved.	N/A

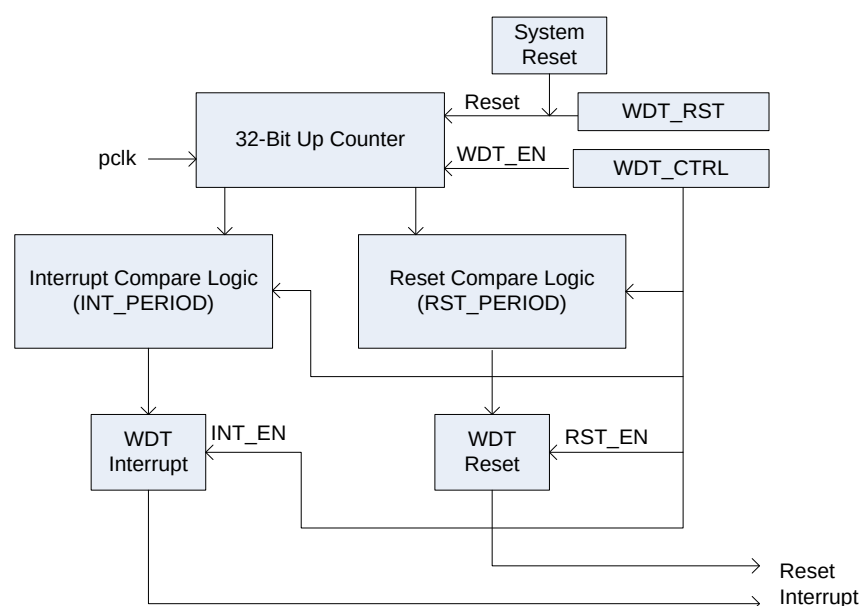
19.0 Watchdog Timer

The watchdog timer protects against corrupt or unreliable software, power faults, and other system-level problems, which may place the microcontroller into unsuitable operating states. When the application is working correctly, application software periodically resets the watchdog counter. If the watchdog timer interrupt is enabled and the software does not reset the counter within the interrupt time period (INT_PERIOD), the watchdog timer generates a watchdog timer interrupt. If the watchdog timer reset is enabled and the software does not reset the counter within the reset time period (RST_PERIOD), the watchdog timer generates a system reset.

The watchdog timer features sixteen programmable time delay periods, 2^{16} through 2^{31} clock cycles for both the interrupt and reset periods.

The watchdog timer counter, which is not directly accessible, is reset on all forms of reset.

Figure 19-1. Watchdog Timer Block Diagram



19.1 Enabling and Disabling the Watchdog Timer

The application software must set the WDT_EN bit to enable the watchdog timer. Because the watchdog timer is free-running, the following procedure must be followed when enabling the watchdog timer to prevent an unintended reset during the enable process.

1. Write 0x0000_00A5 to WDT_RST
2. Write 0x0000_005A to WDT_RST
3. Set WDT_EN bit

The watchdog timer can be disabled in two ways:

- The application software can clear WDT_EN bit to “0” or,
- By a POR, which clears WDT_EN to “0”.

The interrupt and reset signals from the watchdog timer can also be enabled or disabled independently through the INT_EN and RST_EN bits of the WDT_CTRL register.

19.2 Selecting the Time Delay Period

Two independent time delay periods may be specified in the watchdog timer:

- **INT_PERIOD**—Specify the time delay before the watchdog timer sends an interrupt to the CPU.
- **RST_PERIOD**—Specify the time between the reset of the time delay before the watchdog timer sends a system reset request. In a correctly operating system, application software resets the watchdog timer in the watchdog timer interrupt routine before the watchdog counter exceeds the reset period. So, the reset never occurs.

INT_PERIOD must be less than RST_PERIOD or the interrupt never occurs. The interrupt routine must reset the watchdog counter by writing 0x0000_00A5 then 0x0000_005A to the WDT_RST register before the RST_PERIOD or the watchdog timer generates a system interrupt, if enabled.

There are sixteen choices of time delay periods for the watchdog timer: 2^{16} through 2^{31} PCLK cycles. The time delay for a specific clock source is calculated as follows:

Interrupt Period = Clock Source Period $\times$ WDT_CTRL.INT_PERIOD

Reset Period = Clock Source Period $\times$ WDT_CTRL.RST_PERIOD

Examples of approximate interrupt period for different watchdog timer INT_PERIOD are listed in Table 19-1.

Table 19-1. Watchdog Timer Interrupt Period Example

Clock Source	INT_PERIOD	Interrupt Period
120MHz PCLK	1111 (2^{16})	546 μ s
120MHz PCLK	1011 (2^{20})	8.5ms
120MHz PCLK	0110 (2^{25})	280ms
120MHz PCLK	0000 (2^{31})	16.75s

19.3 Watchdog Timer Operation

Utilizing the watchdog timer in the application software is straightforward. As early as possible in the application software, enable the watchdog timer interrupt and watchdog timer reset. Periodically the application software must write to the WDT_RST register to reset the watchdog counter. If program execution has become lost, the watchdog timer interrupt will occur, giving the system a “last chance” to recover from whatever circumstance caused the improper code execution. The interrupt routine can either attempt to repair the situation or allow the watchdog timer reset to occur. In the event of a system software failure, the interrupt will not be executed and the watchdog system reset will recover operation.

At the earliest possible point in the reset routine in the application software, the RST_FLAG bit should be interrogated to see if the reset was caused by the watchdog timer. If so, the application software should assume there was a program execution error and take whatever steps necessary to guard against a software corruption issue.

19.4 Watchdog Timer Registers

Address assignments for registers are outlined below.

Table 19-2. Watchdog Timer Register Addresses

Offset Address	Access	Register	Register Set
0x0000	RW	WDT_CTRL	Watchdog Timer Control Register
0x0004	RW	WDT_RST	Watchdog Timer Reset Register

Refer to section 3.0 Digital Peripherals and System Control Registers for details on the reset values and various access modes.

19.4.1 Watchdog Timer Control Register (WDT_CTRL, Offset 0x0000)

This register is reset only by a POR and is unaffected by all other reset.

Table 19-3. WDT_CTRL (Offset 0x0000)

Position	31	30	29	28	27	26	25	24
Field	RST_FLAG	RFU[6:0]						
POR	0	000 0000						
Reset	s	000 0000						
Access	RW	R						
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
POR	0000 0000							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[3:0]				RST_EN	INT_EN	INT_FLAG	WDT_EN
POR	0000				0	0	0	0
Reset	0000				N/A	N/A	N/A	N/A
Access	R				RW	RW	RW	RW
Position	7	6	5	4	3	2	1	0
Field	RST_PERIOD[3:0]				INT_PERIOD[3:0]			
POR	0000				0000			
Reset	N/A				N/A			
Access	RW				RW			

Name	Bits	Description	Settings	
INT_PERIOD	3:0	Watchdog Interrupt Period. The watchdog timer asserts an interrupt, if enabled, if the CPU does not write the watchdog reset sequence to the WDT_RST register before the watchdog timer has counted this time period since the last timer reset.	<u>INT_PERIOD</u>	<u>Clock Cycles</u>
			1111	2 ¹⁶
			1110	2 ¹⁷
			1101	2 ¹⁸
			1100	2 ¹⁹
			1011	2 ²⁰
			1010	2 ²¹
			1001	2 ²²
			1000	2 ²³
			0111	2 ²⁴
			0110	2 ²⁵
			0101	2 ²⁶
			0100	2 ²⁷
			0011	2 ²⁸
			0010	2 ²⁹
			0001	2 ³⁰
			0000	2 ³¹

Name	Bits	Description	Settings
RST_PERIOD	7:4	Watchdog Reset Period. The watchdog timer asserts a reset, if enabled, if the CPU does not write the watchdog reset sequence to the WDT_RST register before the watchdog timer has counted this time period since the last timer reset.	See watchdog interrupt period setting.
WDT_EN	8	Watchdog Timer Enable.	0: Watchdog timer is disabled 1: Watchdog timer is enabled
INT_FLAG	9	Watchdog Timer Interrupt Flag.	0: Watchdog timer interrupt is not pending. 1: Watchdog timer timeout has occurred. An interrupt is pending.
INT_EN	10	Watchdog Timer Interrupt Enable.	0: Watchdog timer interrupt disabled 1: Watchdog timer causes an interrupt if INT_FLAG=1
RST_EN	11	Watchdog Timer Reset Enable.	0: Watchdog timer reset disabled 1: Watchdog timer causes a reset if enabled.
RFU	30:12	Reserved	
RST_FLAG	31	Watchdog Timer Reset Flag.	0: Previous reset was not caused by the watchdog timer 1: Previous reset was caused by the watchdog timer

19.4.2 Watchdog Timer Reset Register (WDT_RST, Offset 0x0004)

Table 19-4. WDT_RST (Offset 0x0004)

Position	31	30	29	28	27	26	25	24
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	15	14	13	12	11	10	9	8
Field	RFU[7:0]							
Reset	0000 0000							
Access	R							
Position	7	6	5	4	3	2	1	0
Field	WDT_RST[7:0]							
Reset	0000 0000							
Access	W							

Name	Bits	Description	Settings
WDT_RST	7:0	Watchdog Timer Reset Register. Writing the watchdog counter 'reset sequence' to this register resets the watchdog counter. If the watchdog count exceeds INT_PERIOD, then a watchdog interrupt occurs, if enabled. If the watchdog count exceeds RST_PERIOD, then a watchdog reset occurs, if enabled.	The reset sequence is 0x0000_00A5 followed by 0x0000_005A. Always reads zero.
RFU	31:8	Reserved.	

20.0 Serial Flash Emulation Interface

The Serial Flash Emulation (SFE) function is to emulate a SPI Flash device. An external SPI Master is used to access various volatile and non-volatile memory. Using the configuration registers, the Arm core can organize the internal memory resources into address maps. The arm core can read and write to these memories. This interface has the following features:

- Up to 50MHz SPI CLK (Fast Read, all widths) and 25MHz Slow Read
- Single READ, Single/Dual/Quad FAST READ commands
- Single/dual/quad RAM Write Commands
- 24/32-bit Address versions of all READ/FAST_READ/WRITE/WRITE
- Responds to READ_STATUS
- Responds to READ_SFDP
- Responds to READ_MFR_ID
- Responds to 4BYTE_ENTER/4BYTE_EXIT opcodes to enable 4/3 byte addressing on normal 3-byte read/write opcodes
- Responds to SOFT_RESET_EN/SOFT_RESET commands and keeps tracks of Reset Sequence
- Ability to capture data on MISO pin
- Supports READ_SFDP (Read Serial Flash Discoverable Parameters) with 256 bytes of storage
- RAM and Flash regions are configurable and defined by RAM/Flash System Base Address and RAM/Flash System Top address
- Supports ability to lock RAM Read or Write accesses from OTP bits or configuration bits. For configuration lock bits, once RAM Read or Write lock bit is set, the corresponding Read/Write Enable cannot be modified until Reset Event occurs.

20.1 Operation

This interface is an AMBA peripheral to support serial communication to external master devices.

The APB interface is used to configure the SFE settings and to read the settings/statuses/interrupts.

The AHB interface is responsible for reading and writing data from and to memory elements. The SFE provides interrupts to the Arm core to manage the block. It also has an SPI Slave Interface block responsible for receiving SPI commands from the external SPI Master, receiving SPI write data, and transmitting SPI read data/status to the external SPI Master. The Command/Address Decoder's function is to decode the SPI commands/address and starting the memory read/write transactions on the AHB bus or starting the process of returning status/ID/flash parameters to the External SPI Master. The Data Control block gathers write data from SPI interface to be send to AHB interface or gathers read data from memory elements or status/ID/flash parameters to return to the SPI Interface block.

The SFE registers are kept separate from the bus interpreter to ease switching bus protocol without needing to modify the registers.

After a SPI command is received, address is checked to ensure it is within the range of the selected memory. Based on the opcode of the command, a read or write request is issued to read or write to the designated memory location.

20.1.1 Pin Configuration

For proper operation of SFE, P0.6 pin must be set to the Alternate Function 2 (SFSPIS_EN) This pin must have a 25kΩ external pullup.

20.1.2 Slave Command/Read/Write Operations

When Slave Select (SFSPIS_SS0) is asserted, and if,

- READ_ID, READ_SPI_STATUS_REG or READ_SFDP is detected, it returns data immediately after the Command phase.
- Memory access is detected, SPI Slave gathers the next 3 or 4 bytes of SPI address based on the opcodes.
 - o For write commands, data is prepared for AHB transaction.
 - o For read commands, it expects data to be ready at the right SCLK and starts to shift out data SPI0_DIO[3:0] based on the Single/Dual/Quad mode.

Table 20-1. SFE Commands and Opcodes

Command	Opcode	Command Description
<i>Read Commands</i>		
READ	0x03	Serial Read (3-byte address)
FAST_READ	0x0B	Serial Fast Read (3-byte address)
4BYTE_READ	0x13	Serial Read (4-byte address)
4BYTE_FAST_READ	0x0C	Serial Fast Read (4-byte address)
4BYTE_DUAL_FAST_READ	0x3C	Dual fast Read (4-byte address)
QUAD_FAST_READ	0x6B	Quad Fast Read (3-byte address)
4BYTE_QUAD_FAST_READ	0x6C	Quad Fast Read (4-byte address)
<i>Write Commands</i>		
WRITE	0x02	Serial Write (3-byte address)
4BYTE_WRITE	0x12	Serial Write (4-byte address)
DUAL_FAST_WRITE	0xA2	Dual Fast Write (3-byte address)
QUAD_FAST_WRTIE	0x32	Quad Fast Write (3-byte address)
4BYTE_QUAD_FAST_WRITE	0x34	Quad Fast Write (4-byte address)
<i>Address Mode Commands</i>		
SOFT_RESET_EN	0x66	Soft Reset Enable
SOFT_RESET	0x99	Soft Reset
4BYTE_ENTER	0xB7	Enter 4-byte address mode
4BYTE_EXIT	0xE9	Exit 4-byte address mode
<i>Informational Commands</i>		
READ_STATUS	0x05	Read Status Register
READ_MFR_ID	0x9F	Read Manufacturing ID
READ_SFDP	0x5A	Read Serial Flash Discoverable Parameters

Note: 3-byte address commands can be used in 4-byte address mode if ENTER_4BYTE command is issued.

After receiving a valid command, if:

- The SPI command is a *read* command, it is checked if the address is targeted for one of the memory elements and read permission is allowed. If the address does not fall into any region of the memory elements, SFE Slave does not drive SDIO pins during the data phase. Similarly, if SPI address falls into a memory region but if read permission is not allowed, the SFE Slave does not drive SPI0_DIO[3:0] pins during the data phase. If SPI address falls into a memory region and read permission is allowed, a memory read transaction occurs.
- The command is a READ_SPI_STATUS, READ_ID or READ_SFDP, the interface immediately generates the correct byte to the SFE Interface block to output on the SPI bus.
- The command is a *write* interface, the SFE SPI interface expects the correct number of address bytes (3 or 4). It is checked if the address is targeted for the SRAM and write permission is allowed. If the address does not fall into any location within the SRAM, the write command is effectively ignored. Similarly, if SPI address falls into the SRAM region, but if the write permission is not allowed, the write command is effectively ignored. If the SPI address falls within the SRAM region and write permission is allowed, it starts a memory write transaction.

20.1.3 Memory Regions

SFE SPI interface can only support one FLASH and one RAM instance. The FLASH is always readable, and RAM is both readable and writable.

FLASH_SBA register (FLASH System Base Address) and RAM_SBA register (RAM System Base Address) are used in conjunction with FLASH_STA register (Flash System Top Address) and RAM_STA register (RAM System Top Address) to define the valid regions for Flash and RAM accesses. If the SPI host tries to read beyond the defined regions, random read data may return to SPI interface. Any writes initiated by SPI host outside of the RAM defined region is blocked.

20.1.4 Flash MPU

Refer to chapter 12.1.9, Flash Encrypted Region for information on the flash MPU and how it is related to the Serial Flash Emulation (SFE) interface.

20.2 SFE Registers

Address assignments for the SFE Interface registers are outlined in Table 20-2.

Table 20-2. SFE Registers Address (Base ADDR= 0x400A 0000)

Offset	Register Name	Access	Description
0x0400	SF_CFG	RW	Serial Flash Configuration Register
0x0404	RFU	R	Reserved
0x0408	RFU	R	Reserved
0x040C	RFU	R	Reserved
0x0410	SFDP_SBA	RW	Serial Flash Discoverable Parameter System Base Address
0x0414	FLASH_SBA	RW	FLASH's System Base Address
0x0418	FLASH_STA	RW	FLASH's System Top Address
0x041C	RAM_SBA	RW	RAM's System Base Address
0x0420	RAM_STA	RW	RAM's System Top Address

20.2.1 SFE Configuration Register (SF_CFG, Offset 0x0400)

Position	31	30	29	28	27	26	25	24
Field	RFU							
Reset	0000 0000							
Access	R							
Position	23	22	21	20	19	18	17	16
Field	RWLOCK	RRLOCK	RFU				WR_EN	RD_EN
Reset	0	0	0000				0	0
Access	RW	RW	R				RW	RW
Position	15	14	13	12	11	10	9	8
Field	FLOCK	RFU						
Reset	0	000 0000						
Access	RW	R						
Position	7	6	5	4	3	2	1	0
Field	RFU							DRLE
Reset	000 0000							0
Access	R							RW

Name	Bits	Description	Settings
RFU	31:24	Reserved	
RWLOCK	23	RAM Write Lock. When RWLOCK = 0, APB can write 1 to the bit. Once it is 1, APB cannot write 0 to the bit or change the WR_EN. When RWLOCK = 1, it only goes back to 0 after a reset event. There is an OTP RAM Write Lock input port to the SFE. If this input is 1, RWLOCK is always 1.	
RRLOCK	22	RAM Read Lock. When RRLOCK = 0, APB can write 1 to the bit. Once it is 1, APB cannot write 0 to the bit or change the RD_EN. When RRLOCK = 1, it only goes back to 0 after a reset event. There is an OTP RAM Read Lock input port to the SFE. If this input is 1, RRLOCK is always 1.	
RFU	22:18	Reserved	
WR_EN	17	RAM Write Enable. When RLOCK = 1, this bit cannot be changed.	
RD_EN	16	RAM Read Enable. When RLOCK = 1, this bit cannot be changed.	
FLOCK	15	Flash Lock. When FLOCK = 0, APB can write 1 to the bit. Once it is 1, APB cannot write 0 to the bit. When FLOCK = 1, it only goes back to 0 after a reset event. There is an OTP Flash Lock input port to the SFE. If this input is 1, FLOCK is always 1. There is no other function associated with this bit.	
RFU	15:1	Reserved	
DRLE	0	Data Rise Launch Edge Enable (default data fall launch edge)	

20.2.2 SFE Discoverable Parameter System Base Register (SFDP_SBA, Offset 0x0410)

Position	31	30	29	28	27	26	25	24
Field	SFDP_SBA[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	SFDP_SBA[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	SFDP_SBA[15:8]							
Reset	0000 0000							
Access	RW							
Position	7	6	5	4	3	2	1	0
Field	RFU							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
SFDP_SBA	31:8	SFDP upper 24 bits' System Base Address	
RFU	7:0	Reserved	

20.2.3 Flash's System Base Address Register (FLASH_SBA, Offset 0x0414)

Position	31	30	29	28	27	26	25	24
Field	FLASH_SBA[31:24]							
Reset	0001 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	FLASH_SBA[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	FLASH_SBA[15:10]						RFU	
Reset	00 0000						00	
Access	RW						R	
Position	7	6	5	4	3	2	1	0
Field	RFU							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
FLASH_SBA	31:10	FLASH upper 22 bits' System Base Address. The address should be within the available FLASH memory address space.	Once PERCKCN1.SFES is set to 1, this register is updated with the corresponding OTP value.
RFU	9:0	Reserved	

20.2.4 Flash's System Top Address Register (FLASH_STA, Offset 0x0418)

Position	31	30	29	28	27	26	25	24
Field	FLASH_STA[31:24]							
Reset	0001 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	FLASH_STA[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	FLASH_STA[15:10]						RFU	
Reset	10 0001						00	
Access	RW						R	
Position	7	6	5	4	3	2	1	0
Field	RFU							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
FLASH_STA	31:10	FLASH upper 22 bits' System Top Address. The address should be within the available FLASH memory address space.	Once PERCKCN1.SFES is set to 1, this register is updated with the corresponding OTP value.
RFU	9:0	Reserved	

20.2.5 RAM's System Base Address Register (RAM_SBA, Offset 0x041C)

Position	31	30	29	28	27	26	25	24
Field	RAM_SBA[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	RAM_SBA[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	RAM_SBA[15:10]						RFU	
Reset	0000 00						00	
Access	RW						R	
Position	7	6	5	4	3	2	1	0
Field	RFU							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
RAM_SBA	31:10	RAM upper 22 bits' System Base Address. The address should be within the available RAM memory address space.	Once PERCKCN1.SFES is set to 1, this register is updated with the corresponding OTP value.
RFU	9:0	Reserved	

20.2.6 RAM's System Top Address Register (RAM_STA, Offset 0x0420)

Position	31	30	29	28	27	26	25	24
Field	RAM_STA[31:24]							
Reset	0000 0000							
Access	RW							
Position	23	22	21	20	19	18	17	16
Field	RAM_STA[23:16]							
Reset	0000 0000							
Access	RW							
Position	15	14	13	12	11	10	9	8
Field	RAM_STA[15:10]						RFU	
Reset	0000 00						00	
Access	RW						R	
Position	7	6	5	4	3	2	1	0
Field	RFU							
Reset	0000 0000							
Access	R							

Name	Bits	Description	Settings
RAM_STA	31:10	RAM upper 22 bits' System Top Address. The address should be within the available RAM memory address space.	Once PERCKCN1.SFES is set to 1, this register is updated with the corresponding OTP value.
RFU	9:0	Reserved	

21.0 Secure ROM

21.1 Introduction

This chapter describes which services its Secure ROM code provide and how it simplifies the security requirements expressed in the most demanding security standards. It also presents the different boot scenarios and all mechanisms especially designed for security compliance.

21.1.1 Notations

Hexadecimal numbers are represented with a 0x in front of them and in Courier New.

Example: 0xF8DF

Decimal numbers are indicated by a 10 in index.

Example: 123₁₀

21.1.2 Endianness

The chosen convention is 'big endian'. Hexadecimal numbers are presented most significant byte first and most significant bit first. It means the 32-bit number 0x123456 is 1193046₁₀.

21.2 Description

The chip automatically checks the integrity of this ROM code at reset and jumps to the beginning of this ROM to start executing its code.

This ROM code can securely:

- Manage the chip lifecycle
- Program the embedded NOR flash
- Program the embedded OTP memory
- Start applications from embedded NOR flash
- Load and run test programs, using a flexible mechanism, using applets loaded in internal RAM.

21.3 Secure Lifecycle

The cryptographic scheme used by the ROM code relies on ECDSA with curve *P*-256, using the FIPS 186-4 ECDSA standard. All instances of 'securely' below means 'authenticated by a digital signature'.

- By default, the chip is secure by the ECC P-256 Maxim Root Key, its public part being hard-coded in the ROM code.
- For unlocking chips, the customer has to generate his own ECC key pair, the Customer Root Key, CRK, and to securely program the public part in the chip OTP (the same key for all chips).
- Once this is done, the chip control is fully given to the customer and controlled by the CRK key.
- The customer may decide to kill the product (i.e., not responding any more), by securely sending a specific command to the ROM code.

21.4 Secure Boot

The ROM code, depending on the OTP configuration, loads, authenticates, and runs the 2nd level application. Therefore, the first customer application runs after the ROM code finishes execution. The digital signature verification guarantees that no illicit application can be run from the SoC.

21.5 Secure Update

The ROM code is capable of securely updating the embedded flash and OTP.

This secure update protocol is also able to program the SoC internal OTP, used for memories and security configuration.

The ROM also allows a mechanism to load applets. This secure update is also able to load and run small programs in the internal RAM. The serial port link is available for the secure download.

21.6 Boot Scenarios

21.6.1 Secure Boot

The secure boot is stored in the internal ROM memory of the secure SoC. This is the code that is always run first when the SoC starts. This initiates the chain of trust that applications execution relies on: the secure boot guarantees to the applications the platform they are running on is trustworthy and is the one expected; the secure boot also guarantees that the applications running on the platform are only the authorized ones, so only trusted applications can run on the platform.

The security the secure boot implements to provide trust is based on standard public-key cryptography. The secure boot uses the customer public key to verify the applications and authenticate the updates.

The objective of the secure boot is to provide high security strength to the platforms while keeping flexibility and ability to adapt to various customer needs. The secure boot thus has multiple mechanisms.

The first mechanism is the verification before launch of user applications from the flash memory: this verification guarantees that only authenticated applications can run on the platform.

The second mechanism is the secure loading in the Secure SoC internal RAM through the Secure Communication Protocol, SCP.

Note: The word 'application' or 2nd level application (SLA) covers any kind of firmware binary run immediately after the secure boot. This application may be a 2nd level loader, a demo standalone application, or even a full OS.

21.6.2 Customer Key

The secure boot performs authentications using the ECDSA P-256 algorithm, with 256-bit ECC keypair and SHA-256 secure hash function.

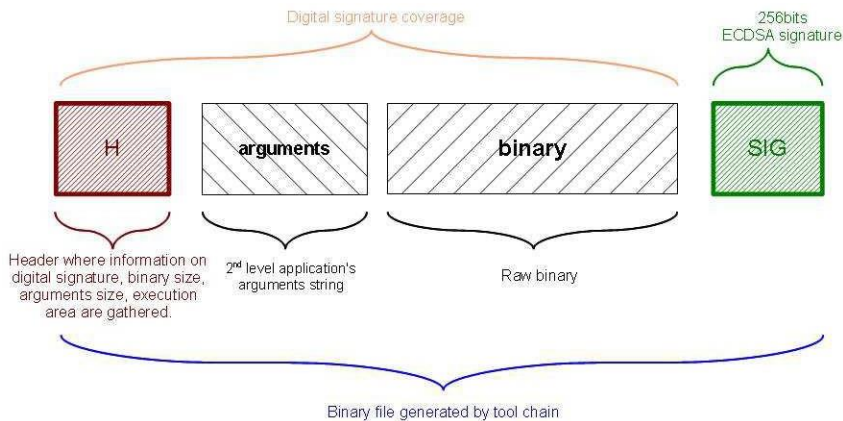
21.6.3 Secure Scenario

The Secure boot starts in secure conditions and performs security controls (including cryptographic self-tests) before verifying the application stored in embedded flash memory.

21.7 Application Images

The SoC internal flash holds the application code. The application image should have the format depicted in Figure 21-1.

Figure 21-1. Application Image Format



- Secure header section
 - Synchronization pattern (64bits): 0x4447444557534948.
 - Format Version Number (32bits). Version of the binary format that shall match what is expected by the ROM code (incremented for each new release of the ROM code). The ROM Code considers a 'reference version' that could be 'older' than current ROM Code one. This is the base version from which the SLA refers (at minimum) to be granted to be run.
 - Raw binary load address (32bits). This address is where binary is going to be copied to.
 - Raw binary length (32bits). This length is the one of binary itself, not binary section
 - Raw binary jump address (32bits). This address is where to start execution.
 - 2nd level application's arguments size (32bits). This is the size of arguments string.
 - Application version number (32bits): this number is used for comparison between the two potential headers found in the two banks. Refer to section 21.10.4 for 'Load & Launch' sequence.
- Arguments section
 - It is where application's arguments are located.
- Application binary section
 - It is where application's raw binary is located.
- Signature section
 - This section holds the 512bits ECDSA signature of binary. This digital signature is generated on Host side. ROM Code role is to verify it. If it matches, execution is granted, and the ROM code jumps into the binary entry point address. If signature does not match, the ROM code turns the platform into the 'shutdown mode'. The digital signature is processed from the synchronization pattern (included) to the end of raw binary.
 - The application secure header contains an absolute jump address, jump-address.
 - Refer to section 21.10.4 for 'Load & Launch' sequence (SLA boot).

21.8 Secure ROM Services

The secure boot is the code always run first at the SoC start up. This initiates the chain of trust that applications execution relies on: the secure boot guarantees to the applications the platform they are running on is trustworthy and is the one expected; the secure boot also guarantees that the applications running on the platform are only the authorized ones, so only trusted applications can run on the platform.

The objective of the secure boot is to provide high security strength to the platforms while keeping flexibility and ability to adapt to various customer needs. The secure boot thus has multiple mechanisms.

The first mechanism is the verification before the launch of user applications: this verification guarantees that only authenticated applications can run on the platform (see section 21.9.5).

The second mechanism is the secure programming of the platform memories (OTP, flash), including applications secure updates. The loading of applets in the internal RAM memory allows running tests programs (see section 21.9.4).

The secure boot authentication mechanism is based on standard public-key cryptography. The SoC secure boot uses customer public key to verify the applications and authenticate the updates (see section 21.8.1).

21.8.1 Customer Key

The SoC secure boot performs authentications using the ECDSA P-256, with 256-bit ECC key pairs and SHA-256 secure hash function. These algorithms require keys values available on the platform.

A small PKI is defined for SoC secure boot security. This PKI is based on a Maxim Integrated Root Authority key, whose public key is stored in the ROM memory. Therefore, any customer key shall be certified by this Root key before use on the platform.

The chips are delivered to customers without any customer key. The first, mandatory step for the customer is to program the certified 256-bit ECC public key in the internal OTP memory (nothing else can be done before this step).

The sequence for programming the customer key is the following:

1. The customer securely generates the ECC key pair CRK, the public part is the customer key to be programmed in SoC OTP.
2. The customer exports the CRK public key and sends it to Maxim Integrated for certification.
3. Maxim Integrated generates a certificate for this CRK public key, using the Maxim Integrated Root Key, the MRK. The CRK certificate contains the CRK public key and a digital signature of this key. Maxim Integrated sends the CRK certificate back to the customer.
4. The customer uploads the CRK certificate to the SoC platform, using the secure boot protocol (Secure Communication Protocol, SCP).
5. The customer resets the chip.

At this stage, the platform secure boot is now able to perform secure updates (see section 21.9) and to launch applications from internal memory (see section 21.9.5), using the customer key, the CRK.

Customer attention is drawn on that the secret part of the ECC key pair CRK shall be handled (stored and used) securely. Maxim Integrated cannot guarantee the chain of trust if this key is compromised. Therefore, Maxim Integrated recommends the use of an HSM or an equivalent hardware device providing physical protection to the secret part of the ECC key pair and complying with the PCI PTS keys protection requirements. Note that Maxim Integrated protects its own MRK secret part by highly secure means, including a high- security grade HSM, compliant with FIPS 140 and PCI PTS security requirements.

21.8.1.1 CRK Certification

- CRK generation: The customer generates own CRK keypair, using PCI PTS compliant tools and procedure: this key is a 256-bit ECC keypair, using the P-256 curve
- PGP key ceremony:
 - A customer identified person (e.g., project manager, security manager, ...) sends the PGP key by email to the Maxim Integrated Business Manager and confirms it through another channel (skype, phone call).
 - The Maxim Integrated Business Manager sends the `crk.certificate` PGP public key by email.
- CRK sending: The customer sends the CRK public key value to Maxim Integrated by email:
 - The CRK is inserted in the body email.
 - The CRK format is:
 - Hexadecimal format: 64 digits using 0123456789ABCDEF digits
 - Respect of the notations described in sections 1.1 and 1.2, i.e. natural format; the number 8903214₁₀ is coded as 87DA2E₁₆

- The email address where to send the public key is:
`crk.certificate@maximintegrated.com`
- The email shall explicitly contain the customer name.
- The email shall be signed by the sender, using PGP.
- CRK certificate reception
 - The sender receives an email signed by the `crk.certificate` PGP key.
 - This email has an attached zip file containing the SCP packets (a set of .packets binary files and a .list file)/
 - These packets are ready for use with the `serial-sender` or `SendSCP` tool. These packets write the CRK and advance the chip lifecycle.

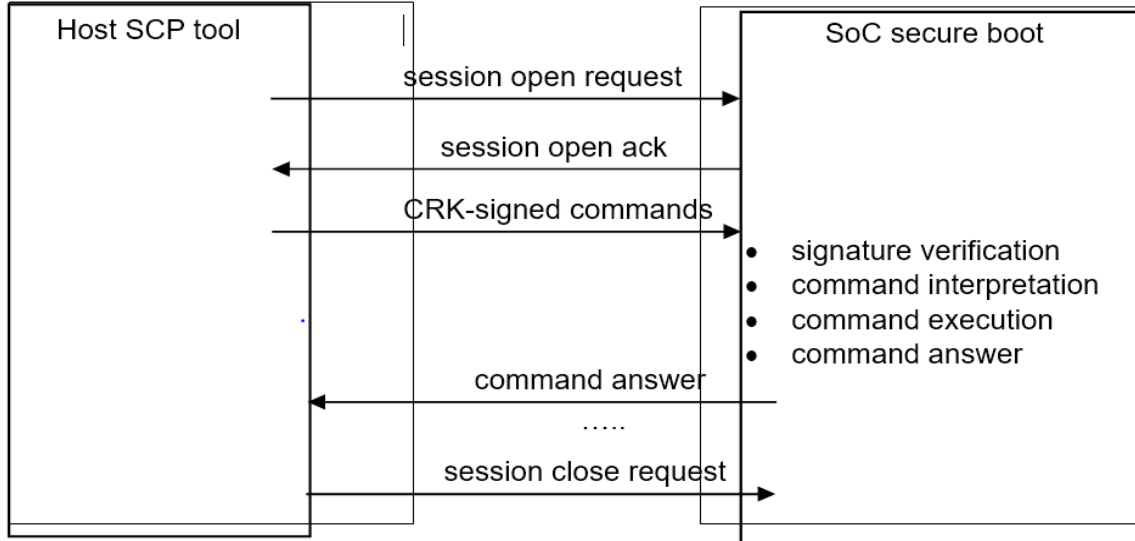
21.9 Secure Loader

A Host, typically a PC or another SoC, can communicate with the SoC secure boot using a secure protocol on the serial link. This protocol, the SCP, enables the Host to send commands to the secure boot. Every command is authenticated by the secure boot before being interpreted and executed. The exceptions are the very first handshake commands. Main commands are those related to memories programming.

21.9.1 SCP Session

The SCP is a session-based protocol. The Host requests session opening to the SoC secure boot and then sends the commands to it. Each command shall be signed using the CRK private key before sending. The digital signature is verified by the SoC secure boot before its interpretation and execution.

Figure 21-2. SCP Session



21.9.2 Packets signature

The commands packets signatures may be computed on-the-fly or before the packets are sent. It enables to split the step for secure handling of the CRK and the step for effective download operation. Therefore, it relaxes some constraints on the security of the manufacturing environment.

21.9.3 OTP Programming

The SoC OTP internal memory is mainly used for platform configuration. As this depends on the customer platform characteristics, it is up to the customer to configure this memory. The SCP provides the ability to program the OTP memory.

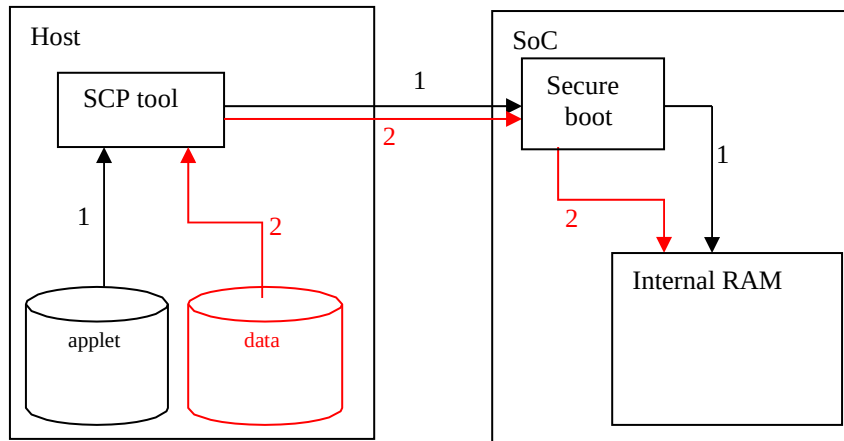
Programming the CRK certificate in OTP is achieved through a dedicated command.

21.9.4 Applets programming

The SCP provides a framework to run secure applets. An applet is a small program securely loaded into the internal RAM and executed directly from internal RAM. The applet can be used for various purposes like testing, programming, etc. This framework is depicted in Figure 21-3:

- The path '1' represents the loading in SoC internal RAM of the applet, using SCP command in a SCP session.
- Once the applet is loaded, the applet can be run from invocation by SCP commands.
- The path '2' represents the data the applet can handle through SCP commands.

Figure 21-3. Application Programming



21.9.5 Secure Boot

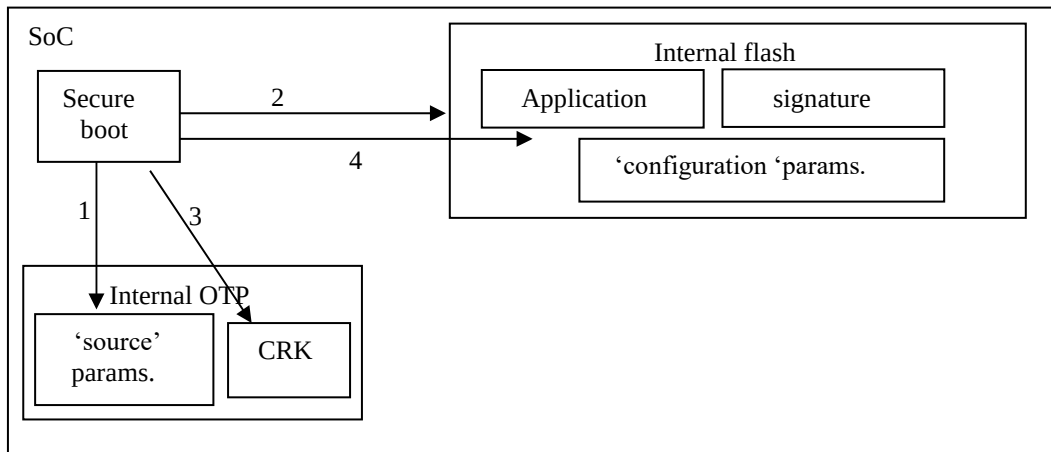
The SoC secure boot verifies any application before launching it. The verification is performed on the application digital signature computed with the CRK private key.

The CRK digital signature is stored within the application in the internal non-volatile memory (e.g., previously programmed by the secure loader mechanism).

Referring to Figure 21-4, the SoC secure boot first retrieves (step '1') configuration parameters stored within the internal OTP memory (previously programmed by the secure loader mechanism). Using these parameters, the SoC secure boot accesses (step '2') the application and its signature. These parameters are called the 'source' parameters as they help the secure boot to access to the application origin, i.e., the non-volatile storage memory.

Once the verification is performed using the CRK public key (step '3'), the secure boot runs the application from the flash (XIP). Parameters for this execution are available in the application header.

Figure 21-4. SoC Secure Application Execution



21.10 Basic Operations

This section describes the steps to perform for running an example application on a SoC. It especially describes how to accomplish the different steps related to the lifecycle and security management.

21.10.1 Configuring the SoC

SoCs are delivered on purpose in a locked state and no application can be run under these conditions. It is up to the customer to unlock the SoCs and make them usable. There are three major steps to achieve that:

- Program the customer ECC-256 public key; this public key is used to check that an authorized, genuine application has been loaded before its execution.
- Set up the user OTP area for secure update link choice.
- Program the signed application in the non-volatile internal memory.

Obviously, each of these three steps includes several smaller steps. They are described in the following sections.

Note that the word ‘application’ covers any kind of firmware binary run immediately after the SoC ROM code. This application may be a 2nd level loader, a demo standalone application, or even a full OS. It mainly depends on customer needs and memories configuration.

21.10.1.1 Program the customer ECC-256 public key

The customer must program the ECC-256 public key on the SoC. This key is written in the customer OTP area and used for application authentication. It must be signed by Maxim before being programmed in the customer OTP.

To achieve this objective, the following steps shall be performed:

1. The customer shall generate a ECC-256 key pair,
 - a. This key is named the CRK, Customer Root Key. Here is an example (32-byte value for the secret, 32-byte for the x affine coordinate, 32-byte value for the y affine coordinate):
 7ac88a77095ce13e593b83904064f98351df9ed430eb143c4abc55a984e57f39
 a823c8857948dc688f3a3ef3f6f220a514f05c2c6c1cef8c9f2f8df11dcf0142
 3be124619cbb51e985328e8e33d321cade19628cc0db43304a7b27f2db8efe
 - b. The key generation and handling shall require the maximum level of security (e.g., generated within a HSM) to comply with security requirements (e.g. PCI PTS, FIPS 140-2, ...)
 - c. During development, it is recommended to use a test key, either the one provided by Maxim Integrated in the Secure ROM package or a key generated by any means: this key need not be securely generated and used.

2. The customer shall send the ECC-256 public part, i.e. the x and y affine coordinates to Maxim Integrated (this step is not required if the customer uses the test key provided in the Secure ROM package, as it is provided already signed):
 - a. The format is ideally a text file, made of two lines, containing a32-byte hexadecimal value of the x affine coordinate and a 32-byte value of the y affine coordinate. Here is an example:
a823c8857948dc688f3a3ef3f6f220a514f05c2c6c1cef8c9f2f8df11dcf0142
3be124619cbb51e985328e8e33d321cade19628cc0db43304a7b27f2db8efe
3. Maxim Integrated signs this public key and sends back the corresponding digital signature using a text file format (i.e., 64-byte hexadecimal value of the digital signature). Maxim Integrated also sends SCP packets (i.e., a set of files) to be downloaded to the SoC.
4. The customer uses the tool `serial-sender` or `SendScp` for SCP packets download.

For further information on the PC host tools, Secure Boot Tools (SBT), contact a Maxim representative.

At this stage, the SoC contains the customer CRK. The device now executes code if and only if it has been signed by the customer private key, i.e., the customer fully controls any download performed on the SoC. The next step is the customer OTP configuration.

21.10.1.2 Updating the CRK

The SoC allows updating the already stored CRK. This feature does not erase and replace, nor overwrite the existing CRK. The process is to program a new CRK through SCP command (REWRITE-CRK) elsewhere into the user's OTP.

- Note that, as for 1st CRK programming, if the area is not 'virgin', programming is going to fail.
- The REWRITE-CRK SCP command has an additional field. This field is for the 'CRK to be replaced' (without its signature). The ROM Code checks then that CRK stored into current area matches the 'CRK to be replaced' to avoid a third party to re-program easily the platform with its own CRK.
 - Before signing the new CRK, Maxim Integrated validates that the entity asking for the signature is the owner of the previous CRK, to be replaced.

21.10.1.3 Choosing the CRK to Use

During SCP communication or SLA digital signature verification, CRK is (verified and) used; but both features do not know about CRK1/2. They only need CRK.

Therefore, the ROM Code always checks:

- CRK2 is (well) programmed into OTP, then CRK2 is CRK.
- CRK2 is not programmed into OTP (or corrupted), then CRK1 is CRK.

21.10.2 Set up the User OTP

The SoC user OTP is not programmed by default.

This OTP configuration is performed using the SCP protocol. As using the CRK private part for SCP packets signatures computation, the key handling shall require the maximum level of security (e.g., generated within a HSM), in order to comply with security requirements (e.g. FIPS 140-2,3). In test conditions, the session-build tool can be used for SCP packets computation, by using the ECDSA CRK test key.

21.10.2.1 User OTP Mapping

The data granularity is one line (8 bytes) in the SoC OTP.

For more details about the session-build tool use (and especially the `write-otp` command), see 21.10.6.2.

Table 21-1. OTP Layout

offset	bit ₆₃	bit ₆₂ .. bit ₁₅	bit ₁₄ ..bit ₀	comments
Userbase (0x10801000) +	Lock bit	data	CV	
0000	1b	CRK ₁ –512bits ECDSA key _(LSB)	XXXX	
...	...	...	...	
0050	1b	'0' from bit ₆₂ to bit ₄₇ and Last CRK₁'s 32bits_(MSB) from bit ₄₆ to bit ₁₅	XXXX	
0058	1b	CRK₁ signature – 512bits _(LSB)	XXXX	
...	...	...	...	
00a8	1b	'0' from bit ₆₂ to bit ₄₇ and Last CRK₁ signature's 32bits_(MSB) from bit ₄₆ to bit ₁₅	XXXX	
00b0	1b	CRK₂ – 512bits ECDSA key _(LSB)	XXXX	
...	...	...	...	
0100	1b	'0' from bit ₆₂ to bit ₄₇ and Last CRK₂'s 32bits_(MSB) from bit ₄₆ to bit ₁₅	XXXX	
0108	1b	CRK₂ signature – 512bits _(LSB)	XXXX	
0158	1b	'0' from bit ₆₂ to bit ₄₇ and Last CRK₂ signature's 32bits_(MSB) from bit ₄₆ to bit ₁₅	XXXX	
0060/HL	1b	'0' from bit ₆₂ to bit ₄₇ and HHA location 32bits from bit ₄₆ to bit ₁₅	xxxx	HHA or Magic Value. If Magic Value, platform goes into shutdown mode
0068	1b	Boot Source ID from bit ₆₂ to bit ₅₅ , '0' from bit ₅₄ to bit ₄₇ and Binary location 32bits from bit ₄₆ to bit ₁₅	xxxx	Byte 5 is Boot Source ID (ignored, always Internal Flash) Byte 3 - Byte 0 is Binary Location address
0200		<i>free to use</i>	XXXX	

21.10.3 Program the Application in Embedded Flash Memory

The application to be run on the SoC is stored in the embedded flash. The handled application binary file is not the file result of the build process, as a specific format is used. The process for application format conversion is described below. For a correct execution, some prerequisites are mandatory:

- The user OTP is correctly configured (see section 21.10.2).
- The application is developed using the 2nd level application (SLA) framework. The framework package includes the documentation, an example of application and the tools for building (Eclipse, gcc tool chain).
- The application has a specific format:
 - It shall be signed with the CRK.
 - The defined application header shall be correctly configured.
 - The header configuration contains information about the binary start address, application version.
 - For these two operations, an access to the CRK private part is needed; the key handling shall require the maximum level of security (e.g., generated within an HSM) to comply with security requirements (e.g., FIPS 140-2). In test conditions, the `ca-sign-build` tool can be used for application signature and header configuration (see 21.10.6.1). The `load_address` and the

`jump_address` can be found in the `.map` file generated at build by the SLA framework or by user's own development environment.

21.10.4 Load & Launch Sequence

In the factory state, the SoC will check flash location `0x10008400` for a valid SLA and reset if one is not found. The address `0x10008400` is the default address divide between the non-encrypted and encrypted flash memory. However, it is recommended to write the OTP value *binary location* to `0x1000C000` to move the SLA address to a flash page boundary to avoid the complications of updating a shared flash page.

21.10.5 Data Files Format

The ROM code host tools handle data (keys, configuration parameters) through the use of files (mainly text files). The main data files formats are provided in this section.

21.10.5.1 ECDSA private key file

The type is `ecdsa-privkey-file`. This text file contains three lines:

- 32-byte hexadecimal-coded secret
- 32-byte hexadecimal-coded x affine coordinate
- 32-byte hexadecimal-coded y affine coordinate

Example:

```
7ac88a77095ce13e593b83904064f98351df9ed430eb143c4abc55a984e57f39
a823c8857948dc688f3a3ef3f6f220a514f05c2c6c1cef8c9f2f8df11dcf0142
3be124619cbb51e985328e8e33d321cade19628cc0db43304a7b27f2db8efe
```

The corresponding decimal value for the secret above is:

```
5553649259032970591845465084231497643775489832492785515079727354204812827013
```

21.10.5.2 ECDSA public key file

The type is `ecdsa-pubkey-file`. This text file contains two lines:

- 32-byte hexadecimal-coded x affine coordinate
- 32-byte hexadecimal-coded y affine coordinate

Example:

```
a823c8857948dc688f3a3ef3f6f220a514f05c2c6c1cef8c9f2f8df11dcf0142
3be124619cbb51e985328e8e33d321cade19628cc0db43304a7b27f2db8efe
```

21.10.5.3 ECDSA signature file

The type is `ecdsa-signature-file`. This text file contains one line:

- 64-byte hexadecimal-coded signature, computed using `ecdsa-p256-sha256`

Example:

```
5b5c553b6405a3bb432d76566e9e480798b27dcda0d69af22cc230df89cf5483
ab99f0a36e0f2024f151243a26186fab13e7e8067d001c98c13d4ebcc3d86bea
```

21.10.5.4 ECDSA signed public key

This file format is the combination of the ECDSA public key file format and ECDSA signature file format. The type is `ecdsa-sign-pubkey-file`.

This text file contains three lines:

- 32-byte hexadecimal-coded x affine coordinate

- 32-byte hexadecimal-coded y affine coordinate
- 64-byte ECDSA signature

Example:

```
a823c8857948dc688f3a3ef3f6f220a514f05c2c6c1cef8c9f2f8df11dcf0142
3be124619cbb51e985328e8e33d321cade19628cc0db43304a7b27f2db8efe
5b5c553b6405a3bb432d76566e9e480798b27dcda0d69af22cc230df89cf5483
ab99f0a36e0f2024f151243a26186fab13e7e8067d001c98c13d4ebcc3d86bea
```

21.10.5.5 Binary File

The type is `binary-file`. This binary file is made of raw bytes.

21.10.5.6 s19 file

The type is `s19-file`.

This text file contains one or multiple lines:

- Only the S3-flagged lines are considered,
- The created s19 files have correct CRC byte at the end of each line,

21.10.6 Tools

The tools are used in a logical sequence.

Except the ones not using any option, every tool has a `.ini` file, which contains the list of the options with already determined values.

21.10.6.1 Customer Application Signature

NAME

`ca_sign_build`: build binary file including ECDSA signature and SLA header from customer application binary file.

SYNOPSIS

```
ca_sign_build [ecdsa=<ecdsa-privkey-file name>] [algo=ecdsa] [ca=<application
binary-file>] [sca=<signed application binary-file>] [jump_address=<4-byte
hexadecimal value>] [arguments=<string>]
```

OPTIONS

These options are those described in the ROM code 2nd level application header format:

- **ecdsa**=<ecdsa-privkey-file name>: 256-bit ECC key used for application signature
- **algo**=ecdsa: This field describes which algorithm has to be used (the tool supports also `rsa` for MAX32590 use)
- **ca**=<application binary file>: The origin (input) binary file
- **sca**=<signed application binary file>: The final (output) signed+header binary file; It contains the:
 - Header
 - Input binary file
 - Appended signature
- **jump_address**=<4-byte hexadecimal value>: The jump address of the binary application
- **arguments**=<string>: The arguments to be provided to the final application through `argc, argv`; the string is limited to 10KB by the tool
- **version**=<4-byte hexadecimal value>: Identifies the ROM code version, i.e., the header version ROM code can support

- **verbose**=<yes|no>: This option states if verbose mode is activated during execution, or not.

LIMITATIONS

- The arguments string length is limited to 10KB

EXIT CODES

- **0**: everything is OK
- **1**: failure

EXAMPLES

- **ini** file:
#lines starting by a # are considered as comments
#the 256-bit ECC key used for application signature
ecdsa=casignk.key
#the origin (input) binary file
ca=appli.bin
#the chosen algorithm
algo=ecdsa
#the final (output) signed+header binary file
#it contains the header, the input binary file
#and the appended signature
sca=appli.sbin
#the jump address of the binary application
jump_address=01020304
#the binary length is automatically computed
#arguments
argv="string of chars"
#arguments length is automatically computed
#verbosity level
verbose=yes
\$./ca_sign_build.exe ecdsa=casignk.key algo=ecdsa ca=appli.bin
sca=appli.sbin load_address=01020304 jump_address=02030405 verbose=yes

21.10.6.2 SCP Session Build

The SCP session tool computes offline the SCP frames corresponding to the provided parameters. This tool is used for several platforms and some options only apply on some of them (e.g., ecdsa_file and flash_size_mb do not apply).

NAME

session_build: Builds offline set of files containing the packets sent during a SCP session between the host and ROM code.

SYNOPSIS

```
session_build [session_mode=<mode>] [algo=ecdsa] [verbose=yes|no]
[output_file=<file-name-radix>] [pp=<protection-profile>] [ecdsa_file=ecdsa-
privkey-file] [script_file=<file name>] [addr_offset=<address>]
[chunk_size=<byte size>]
```

OPTIONS

- **session_mode**=<mode>: This option states the SCP session mode, i.e., SCP_ANGELA_ECDSA
- **verbose**=<yes|no>: This option states if verbose mode is activated during execution, or not
- **output_file**=<file-name-radix>: This option states the radix for the generated files (packets and logfile)
- **pp**=<protection-profile>: This option states the protection profile, which is ECDSA

- **ecdsa_file**=<ecdsa-privkey-file >: This option states the ECC private key file name
- **script_file**=<script file name>: This option states the script file name to be used for the generated SCP session
 - Scripts syntax is the following:
 - Lines starting by a '#' are considered as comments and are not processed
 - Each line contains only one SCP command
 - Some commands require one parameter (e.g., value, file name)
 - The list of available commands is the following:
 - **write-file <s19-file>**: This command implicitly performs an accurate erase before programming, this command implicitly performs a verify after programming
 - **write-only <s19-file>**: This command performs only the data write operation (no erase before, no verify after)
 - **erase-data <start-addr> <length>**: This command performs an erase on the memory area defined by the 'start-addr' address and the 'length' number of bytes
 - **write-otp <hex offset:2bytes> <hex data value>**
 - **write-crk <ecdsa-signed-pubkey-file>**
 - **write-timeout <target char> <hex timeout value:ms>**: This command sets the timeout value up in milliseconds, for the targeted interface, either UART0 (char '0') or USB ('U') or VBus Detect ('V')
 - **kill-chip**
 - **execute-code <hex address:4bytes>**
- **addr_offset**=<address>: This option states the initial offset for s19 file download, in internal memory
- **chunk_size**=<byte size>: This option sets up the size of the payload buffer used for write-data packets

OUTPUTS

- The output_file radix value is used to generate:
 - A text log file containing the details of each packet, named output_file_radix.log
 - A binary file containing the bytes for each packet:
 - A packet issued by the host is tagged .host., named output_file_radix.host.<file-number>.<action>.packet
 - A packet issued by the bootloader is tagged .bl., named output_file_radix.bl.<file-number>.<action>.packet
 - where <file-number> is an integer value incremented by one for each new binary file created and <action> corresponds to the kind of operations represented by the binary frame (e.g., connection request, hello reply, write mem, ...)

LIMITATIONS

- Every SCP command is not implemented in the tool> Refer to the script available commands list
- The addresses used in the s19 file shall be based on the 0x00000000 value
- The chunk_size value shall be 4094/15354
- The script-file shall end with an end-of-line, i.e., the last command shall be entered with a CR

EXIT CODES

- **0**: everything is OK

- 1: failure.

NOTES

- This tool can be used for other parts (e.g., MAX325xx, MAXQ1852), with other configuration parameters; only those relevant for this SoC are described here

EXAMPLES

```
$cat script.txt
write-file myapplication.s19 write-otp 000A 040506070809
write-bpk 0004 0102030405060708090A0B0C0D0E0F00
write-crk crk_test.signpub write-timeout 0 012C
execute-code 01020304
```

```
$session_build session_mode=SCP_ANGELA_ECDSA ecdsa_file=crk.key verbose=yes
output_file=session script=script.txt addr_offset=00000000 chunk_size=4094
```

The log file session.log and packets files
session.host.1.connection_request.packet,
session.bl.2.connection_reply.packet, session.host.3.ack.packet,
session.host.4.hello_request.packet, ..., are created.

21.10.6.3 Packets serial sender

The serial_sender tool uses packets generated by the session_build tool to send the host. Ones to the bootloader on the serial port and to wait for the .bl. ones.

NAME

serial_sender sends signed packets to the bootloader on the serial link. This tool can be used as a test tool for bootloader and validation tool. FILENAME contains both sent and received packets. Those last ones are used for verification.

SYNOPSIS

Usage: serial_sender [options] FILENAME

OPTIONS

--version show program's version number and exit
-h, --help show this help message and exit
-s SERIAL, --serial=SERIAL define the serial port to use
-v, --verbose enable verbose mode
-l, --license display license
--list-serial display available serial ports
-b, --bl-emulation emulate the bootloader
-t SECONDS define the timeout used for the commands' answers from the device, expressed in seconds. Default value is 35s. This value shall be adapted in function of the flash area to be erased, as this erasure may take some time (so, the timeout shall not be too short).

LIMITATIONS

- The FILENAME contains the list of packets files to be processed. The files are processed by the tool following the sequential order. So, the files shall be correctly sorted at file list building (see the example below).
- The tool sends packets only once. So, the connection request is sent only once. Therefore, the platform shall be reset before starting the tool (and correctly synchronized).

ERRORS

- The `serial_sender` checks the packets received from the bootloader and raises an error if the data is different than the ones expected, except for the `HELLO_REPLY` packet (as it may vary on the USN).

EXAMPLE

```
$ls session.*packet|sort -t '.' -n +1 >session.list
or
$ls session.*packet|sort -t '.' -n -k1 >session.list
$serial_sender --list-serial
COM1
COM3
COM8
$serial_sender -s COM1 session.list -v
```

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	2/21	Initial release	—