

Security Reference Manual for the i.MX RT1050 Processor

Document Number: IMXRT1050SRM
Rev. 1, 04/2018



Contents

Section number	Title	Page
Chapter 1 Security Overview		
1.1	Chapter overview.....	37
1.2	Feature summary.....	37
1.3	High-Assurance Boot (HAB).....	39
1.3.1	HAB process flow.....	40
1.3.2	HAB feature summary.....	42
1.4	Secure Non-Volatile Storage (SNVS) module.....	42
1.4.1	SNVS architecture.....	43
1.5	Data Co-Processor (DCP).....	44
1.6	Standalone True Random Number Generator (TRNG).....	44
1.7	OCOTP_CTRL.....	44
1.8	Central Security Unit (CSU).....	45
1.9	System JTAG Controller (SJC).....	45
1.9.1	Scan protection.....	46
1.10	Bus Encryption Engine (BEE).....	46
Chapter 2 Security System Integration		
2.1	Master ID allocation.....	49
2.2	System-level SNVS connections.....	49
2.2.1	SNVS clock tamper input.....	49
2.2.2	System security violation alarm signals monitored by SNVS.....	50
2.3	Security access error.....	50
2.4	OCRAM TrustZone support.....	50
2.5	Watchdog mechanism.....	51
2.6	Security configuration.....	52
2.7	Field return for retest procedure.....	53

Chapter 3

Section number	Title	Page
	System Boot	
3.1	Chip-specific Boot Information.....	57
3.2	Overview.....	60
3.3	Boot modes.....	61
3.3.1	Boot mode pin settings.....	62
3.3.2	High-level boot sequence.....	62
3.3.3	Boot From Fuses mode (BOOT_MODE[1:0] = 00b).....	63
3.3.4	Serial Downloader (BOOT_MODE[1:0] = 01b).....	64
3.3.5	Internal Boot mode (BOOT_MODE[1:0] = 0b10).....	64
3.3.6	Boot security settings.....	65
3.4	Device configuration.....	66
3.4.1	Boot eFUSE descriptions.....	66
3.4.2	GPIO boot overrides.....	67
3.4.3	Device Configuration Data (DCD).....	68
3.5	Device initialization.....	68
3.5.1	Internal ROM/RAM memory map.....	69
3.5.2	Boot block activation	69
3.5.3	Clocks at boot time.....	70
3.5.4	Enabling Caches.....	72
3.5.5	Exception handling.....	72
3.5.6	Interrupt handling during boot.....	73
3.5.7	Persistent bits.....	73
3.6	Boot devices (internal boot).....	73
3.6.1	Serial NOR Flash Boot via FlexSPI.....	74
3.6.1.1	Serial NOR eFUSE Configuration.....	74
3.6.1.2	FlexSPI Serial NOR Flash Boot Operation.....	75
3.6.1.3	FlexSPI NOR boot flow chart.....	75
3.6.2	Encrypted XIP and Flash Access Protection.....	76
3.6.2.1	Encrypted XIP on Serial NOR via FlexSPI Interface.....	76

Section number	Title	Page
3.6.2.2	FlexSPI NOR Image layout.....	77
3.6.2.3	Key Info Block.....	78
3.6.2.4	Protection Region Descriptor Block (PRDB).....	79
3.6.2.5	Encrypted XIP Flow.....	80
3.6.2.6	Data and Key Endianness.....	81
3.6.3	Serial NAND Flash Boot over FlexSPI.....	82
3.6.3.1	Serial NAND eFUSE Configuration.....	82
3.6.3.2	FlexSPI NAND Flash Boot Flow and Boot Control Blocks (BCB).....	83
3.6.3.3	Firmware Configuration Block.....	87
3.6.3.4	Discovered Bad Blocks Table (DBBT).....	88
3.6.3.5	Bad block handling in ROM.....	89
3.6.4	Serial NOR and NAND configuration based on FlexSPI interface.....	89
3.6.4.1	FlexSPI Configuration Block	89
3.6.4.2	Serial NOR configuration block (512 bytes).....	94
3.6.4.3	Serial NAND configuration block (512 bytes).....	95
3.6.5	Parallel NOR flash Boot over SEMC.....	97
3.6.5.1	Parallel NOR eFUSE Configuration.....	97
3.6.5.2	SEMC Parallel NOR Flash Boot Operation.....	99
3.6.6	Parallel NAND flash Boot over SEMC.....	99
3.6.6.1	Parallel NAND eFUSE Configuration.....	99
3.6.6.2	Parallel NAND Flash Boot Control Blocks (BCB).....	101
3.6.6.3	Firmware Configuration Block (FCB).....	102
3.6.6.4	Discovered Bad Blocks Table (DBBT).....	103
3.6.6.5	Bad block handling in ROM.....	104
3.6.7	Parallel NOR and NAND configuration based on SEMC interface.....	104
3.6.7.1	SEMC Configuration Block.....	104
3.6.7.2	Parallel NOR Configuration Block (80 bytes).....	107
3.6.7.3	Parallel NAND Configuration Block (256 bytes).....	108
3.6.8	Expansion device.....	109

Section number	Title	Page
3.6.8.1	Expansion device eFUSE configuration.....	109
3.6.8.2	MMC and eMMC boot.....	112
3.6.8.3	SD, eSD, and SDXC.....	121
3.6.8.4	IOMUX configuration for SD/MMC.....	121
3.6.8.5	Redundant boot support for expansion device.....	122
3.6.9	Serial NOR/EEPROM through LP SPI.....	124
3.6.9.1	Serial NOR/EEPROM eFUSE configuration.....	124
3.7	Program image.....	125
3.7.1	Image Vector Table and Boot Data.....	125
3.7.1.1	Image vector table structure.....	126
3.7.1.2	Boot data structure.....	127
3.7.2	Device Configuration Data (DCD).....	127
3.7.2.1	Write data command.....	128
3.7.2.2	Check data command.....	130
3.7.2.3	NOP command.....	131
3.7.2.4	Unlock command.....	131
3.8	Plugin image.....	132
3.9	Serial Downloader.....	133
3.9.1	USB.....	135
3.9.1.1	USB configuration details.....	135
3.9.1.2	IOMUX configuration for USB.....	136
3.9.2	Serial Download Protocol (SDP).....	136
3.9.2.1	SDP commands.....	137
3.9.2.1.1	READ_REGISTER.....	137
3.9.2.1.2	WRITE_REGISTER.....	138
3.9.2.1.3	WRITE_FILE.....	138
3.9.2.1.4	ERROR_STATUS.....	140
3.9.2.1.5	DCD_WRITE.....	140
3.9.2.1.6	JUMP_ADDRESS.....	141

Section number	Title	Page
3.10	Recovery devices.....	142
3.11	SD/MMC manufacture mode.....	142
3.12	High-Assurance Boot (HAB).....	143
3.12.1	HAB API vector table addresses.....	144
	Chapter 4 Fusemap	
4.1	Boot Fusemap.....	147
4.2	Lock Fusemap.....	158
4.3	Fusemap Descriptions Table.....	159
	Chapter 5 Central Security Unit (CSU)	
5.1	Overview.....	175
5.1.1	Features.....	175
5.2	Functional description.....	175
5.2.1	Peripheral access policy.....	176
5.2.2	Initialization policy.....	176
5.3	Programmable Registers.....	177
5.3.1	Config security level register (CSU_CSL n).....	178
5.3.2	HP0 register (CSU_HP0).....	182
5.3.3	Secure access register (CSU_SA).....	185
5.3.4	HPCONTROL0 register (CSU_HPCONTROL0).....	188
	Chapter 6 Bus Encryption Engine (BEE)	
6.1	Introduction.....	191
6.1.1	Interface and signal list.....	191
6.2	BEE u-architecture.....	194
6.2.1	BEE_CTRL.....	194
6.2.2	Command Validity Checker (CVC).....	194
6.2.3	Decryption Engine (DE).....	195
6.3	BEE internal registers.....	198

Section number	Title	Page
6.3.1	BEE Control Register (BEE_CTRL).....	200
6.3.2	BEE_ADDR_OFFSET0.....	202
6.3.3	BEE_ADDR_OFFSET1.....	203
6.3.4	BEE_AES_KEY0_W0.....	203
6.3.5	BEE_AES_KEY0_W1.....	203
6.3.6	BEE_AES_KEY0_W2.....	204
6.3.7	BEE_AES_KEY0_W3.....	204
6.3.8	BEE_STATUS.....	205
6.3.9	BEE_CTR_NONCE0_W0.....	205
6.3.10	BEE_CTR_NONCE0_W1.....	206
6.3.11	BEE_CTR_NONCE0_W2.....	206
6.3.12	BEE_CTR_NONCE0_W3.....	206
6.3.13	BEE_CTR_NONCE1_W0.....	207
6.3.14	BEE_CTR_NONCE1_W1.....	207
6.3.15	BEE_CTR_NONCE1_W2.....	207
6.3.16	BEE_CTR_NONCE1_W3.....	208
6.3.17	BEE_REGION1_TOP.....	208
6.3.18	BEE_REGION1_BOT.....	208
6.4	BEE program model.....	209

Chapter 7 **Data Co-Processor (DCP)**

7.1	Overview.....	213
7.1.1	DCP limitations for software.....	215
7.2	Clocks.....	216
7.3	Operation.....	216
7.3.1	Memory copy, blit, and fill functionality.....	217
7.3.2	Advanced Encryption Standard (AES).....	217
7.3.2.1	Key storage.....	217
7.3.2.2	AES OTP key.....	218

Section number	Title	Page
7.3.2.3	Encryption modes.....	218
7.3.3	Hashing.....	220
7.3.4	One-Time Programmable (OTP) key.....	221
7.3.5	Managing DCP channel arbitration and performance.....	221
7.3.5.1	DCP arbitration.....	221
7.3.5.2	Channel-recovery timers.....	222
7.3.6	Programming channel operations.....	223
7.3.6.1	Virtual channels.....	223
7.3.6.2	Context switching.....	224
7.3.6.3	Working with semaphores.....	225
7.3.6.4	Work packet structure.....	226
7.3.6.4.1	Next command address field.....	226
7.3.6.4.2	Control0 field.....	227
7.3.6.4.3	Control1 field.....	229
7.3.6.4.4	Source buffer.....	229
7.3.6.4.5	Destination buffer.....	230
7.3.6.4.6	Buffer size field.....	230
7.3.6.4.7	Payload pointer.....	231
7.3.6.4.8	Status.....	231
7.3.6.4.9	Payload.....	232
7.3.7	Programming DCP functions.....	233
7.3.7.1	Basic memory copy programming example.....	233
7.3.7.2	Basic hash operation programming example.....	234
7.3.7.3	Basic cipher operation programming example.....	236
7.3.7.4	Multi-buffer scatter/gather cipher and hash operation programming example.....	237
7.4	DCP memory map/register definition.....	240
7.4.1	DCP control register 0 (DCP_CTRL).....	241
7.4.2	DCP status register (DCP_STAT).....	243
7.4.3	DCP channel control register (DCP_CHANNELCTRL).....	246

Section number	Title	Page
7.4.4	DCP capability 0 register (DCP_CAPABILITY0).....	247
7.4.5	DCP capability 1 register (DCP_CAPABILITY1).....	248
7.4.6	DCP context buffer pointer (DCP_CONTEXT).....	249
7.4.7	DCP key index (DCP_KEY).....	249
7.4.8	DCP key data (DCP_KEYDATA).....	250
7.4.9	DCP work packet 0 status register (DCP_PACKET0).....	251
7.4.10	DCP work packet 1 status register (DCP_PACKET1).....	252
7.4.11	DCP work packet 2 status register (DCP_PACKET2).....	255
7.4.12	DCP work packet 3 status register (DCP_PACKET3).....	256
7.4.13	DCP work packet 4 status register (DCP_PACKET4).....	256
7.4.14	DCP work packet 5 status register (DCP_PACKET5).....	257
7.4.15	DCP work packet 6 status register (DCP_PACKET6).....	257
7.4.16	DCP channel 0 command pointer address register (DCP_CH0CMDPTR).....	258
7.4.17	DCP channel 0 semaphore register (DCP_CH0SEMA).....	259
7.4.18	DCP channel 0 status register (DCP_CH0STAT).....	260
7.4.19	DCP channel 0 options register (DCP_CH0OPTS).....	262
7.4.20	DCP channel 1 command pointer address register (DCP_CH1CMDPTR).....	263
7.4.21	DCP channel 1 semaphore register (DCP_CH1SEMA).....	264
7.4.22	DCP channel 1 status register (DCP_CH1STAT).....	265
7.4.23	DCP channel 1 options register (DCP_CH1OPTS).....	267
7.4.24	DCP channel 2 command pointer address register (DCP_CH2CMDPTR).....	268
7.4.25	DCP channel 2 semaphore register (DCP_CH2SEMA).....	269
7.4.26	DCP channel 2 status register (DCP_CH2STAT).....	270
7.4.27	DCP channel 2 options register (DCP_CH2OPTS).....	272
7.4.28	DCP channel 3 command pointer address register (DCP_CH3CMDPTR).....	273
7.4.29	DCP channel 3 semaphore register (DCP_CH3SEMA).....	274
7.4.30	DCP channel 3 status register (DCP_CH3STAT).....	275
7.4.31	DCP channel 3 options register (DCP_CH3OPTS).....	277
7.4.32	DCP debug select register (DCP_DBGSELECT).....	277

Section number	Title	Page
7.4.33	DCP debug data register (DCP_DBGDATA).....	278
7.4.34	DCP page table register (DCP_PAGETABLE).....	278
7.4.35	DCP version register (DCP_VERSION).....	279

Chapter 8 On-Chip OTP Controller (OCOTP_CTRL)

8.1	Overview.....	281
8.1.1	Features.....	281
8.2	Clocks.....	281
8.3	Top-Level Symbol and Functional Overview.....	282
8.3.1	Operation.....	282
8.3.1.1	Shadow Register Reload.....	283
8.3.1.2	Fuse and Shadow register read.....	283
8.3.1.3	Fuse and Shadow Register Writes.....	284
8.3.1.4	Write Postamble.....	285
8.3.2	Fuse Shadow Memory Footprint.....	286
8.3.3	OTP Read/Write Timing Parameters.....	286
8.3.4	Hardware Visible Fuses.....	288
8.3.5	Behavior During Reset.....	288
8.3.6	Secure JTAG control.....	288
8.4	Fuse Map.....	289
8.5	OCOTP Memory Map/Register Definition.....	289
8.5.1	OCOTP register descriptions.....	289
8.5.1.1	OCOTP Memory map.....	289
8.5.1.2	OTP Controller Control Register (HW_OCOTP_CTRL).....	291
8.5.1.2.1	Offset.....	291
8.5.1.2.2	Function.....	291
8.5.1.2.3	Diagram.....	292
8.5.1.2.4	Fields.....	292
8.5.1.3	OTP Controller Control Register (HW_OCOTP_CTRL_SET).....	293

Section number	Title	Page
8.5.1.3.1	Offset.....	293
8.5.1.3.2	Function.....	293
8.5.1.3.3	Diagram.....	293
8.5.1.3.4	Fields.....	294
8.5.1.4	OTP Controller Control Register (HW_OCOTP_CTRL_CLR).....	295
8.5.1.4.1	Offset.....	295
8.5.1.4.2	Function.....	295
8.5.1.4.3	Diagram.....	295
8.5.1.4.4	Fields.....	295
8.5.1.5	OTP Controller Control Register (HW_OCOTP_CTRL_TOG).....	296
8.5.1.5.1	Offset.....	296
8.5.1.5.2	Function.....	296
8.5.1.5.3	Diagram.....	297
8.5.1.5.4	Fields.....	297
8.5.1.6	OTP Controller Timing Register (HW_OCOTP_TIMING).....	298
8.5.1.6.1	Offset.....	298
8.5.1.6.2	Function.....	298
8.5.1.6.3	Diagram.....	298
8.5.1.6.4	Fields.....	298
8.5.1.7	OTP Controller Write Data Register (HW_OCOTP_DATA).....	299
8.5.1.7.1	Offset.....	299
8.5.1.7.2	Function.....	299
8.5.1.7.3	Diagram.....	299
8.5.1.7.4	Fields.....	299
8.5.1.8	OTP Controller Write Data Register (HW_OCOTP_READ_CTRL).....	300
8.5.1.8.1	Offset.....	300
8.5.1.8.2	Function.....	300
8.5.1.8.3	Diagram.....	300
8.5.1.8.4	Fields.....	300

Section number	Title	Page
8.5.1.9	OTP Controller Read Data Register (HW_OCOTP_READ_FUSE_DATA).....	301
8.5.1.9.1	Offset.....	301
8.5.1.9.2	Function.....	301
8.5.1.9.3	Diagram.....	301
8.5.1.9.4	Fields.....	301
8.5.1.10	Sticky bit Register (HW_OCOTP_SW_STICKY).....	302
8.5.1.10.1	Offset.....	302
8.5.1.10.2	Function.....	302
8.5.1.10.3	Diagram.....	302
8.5.1.10.4	Fields.....	302
8.5.1.11	Software Controllable Signals Register (HW_OCOTP_SCS).....	303
8.5.1.11.1	Offset.....	303
8.5.1.11.2	Function.....	303
8.5.1.11.3	Diagram.....	303
8.5.1.11.4	Fields.....	304
8.5.1.12	Software Controllable Signals Register (HW_OCOTP_SCS_SET).....	304
8.5.1.12.1	Offset.....	304
8.5.1.12.2	Function.....	304
8.5.1.12.3	Diagram.....	304
8.5.1.12.4	Fields.....	305
8.5.1.13	Software Controllable Signals Register (HW_OCOTP_SCS_CLR).....	305
8.5.1.13.1	Offset.....	305
8.5.1.13.2	Function.....	305
8.5.1.13.3	Diagram.....	305
8.5.1.13.4	Fields.....	306
8.5.1.14	Software Controllable Signals Register (HW_OCOTP_SCS_TOG).....	306
8.5.1.14.1	Offset.....	306
8.5.1.14.2	Function.....	306
8.5.1.14.3	Diagram.....	306

Section number	Title	Page
8.5.1.14.4	Fields.....	307
8.5.1.15	OTP Controller Version Register (HW_OCOTP_VERSION).....	307
8.5.1.15.1	Offset.....	307
8.5.1.15.2	Function.....	307
8.5.1.15.3	Diagram.....	307
8.5.1.15.4	Fields.....	307
8.5.1.16	OTP Controller Timing Register 2 (HW_OCOTP_TIMING2).....	308
8.5.1.16.1	Offset.....	308
8.5.1.16.2	Function.....	308
8.5.1.16.3	Diagram.....	308
8.5.1.16.4	Fields.....	308
8.5.1.17	Value of OTP Bank0 Word0 (Lock controls) (HW_OCOTP_LOCK).....	309
8.5.1.17.1	Offset.....	309
8.5.1.17.2	Function.....	309
8.5.1.17.3	Diagram.....	309
8.5.1.17.4	Fields.....	310
8.5.1.18	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG0) ..	311
8.5.1.18.1	Offset.....	311
8.5.1.18.2	Function.....	311
8.5.1.18.3	Diagram.....	312
8.5.1.18.4	Fields.....	312
8.5.1.19	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG1) ..	312
8.5.1.19.1	Offset.....	312
8.5.1.19.2	Function.....	312
8.5.1.19.3	Diagram.....	312
8.5.1.19.4	Fields.....	313
8.5.1.20	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG2) ..	313
8.5.1.20.1	Offset.....	313
8.5.1.20.2	Function.....	313

Section number	Title	Page
8.5.1.20.3	Diagram.....	313
8.5.1.20.4	Fields.....	314
8.5.1.21	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG3)..	314
8.5.1.21.1	Offset.....	314
8.5.1.21.2	Function.....	314
8.5.1.21.3	Diagram.....	314
8.5.1.21.4	Fields.....	315
8.5.1.22	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG4)..	315
8.5.1.22.1	Offset.....	315
8.5.1.22.2	Function.....	315
8.5.1.22.3	Diagram.....	315
8.5.1.22.4	Fields.....	316
8.5.1.23	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG5)..	316
8.5.1.23.1	Offset.....	316
8.5.1.23.2	Function.....	316
8.5.1.23.3	Diagram.....	316
8.5.1.23.4	Fields.....	317
8.5.1.24	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG6)..	317
8.5.1.24.1	Offset.....	317
8.5.1.24.2	Function.....	317
8.5.1.24.3	Diagram.....	317
8.5.1.24.4	Fields.....	318
8.5.1.25	Value of OTP Bank1 Word0 (Memory Related Info.) (HW_OCOTP_MEM0).....	318
8.5.1.25.1	Offset.....	318
8.5.1.25.2	Function.....	318
8.5.1.25.3	Diagram.....	318
8.5.1.25.4	Fields.....	319
8.5.1.26	Value of OTP Bank1 Word1 (Memory Related Info.) (HW_OCOTP_MEM1).....	319
8.5.1.26.1	Offset.....	319

Section number	Title	Page
8.5.1.26.2	Function.....	319
8.5.1.26.3	Diagram.....	319
8.5.1.26.4	Fields.....	320
8.5.1.27	Value of OTP Bank1 Word2 (Memory Related Info.) (HW_OCOTP_MEM2).....	320
8.5.1.27.1	Offset.....	320
8.5.1.27.2	Function.....	320
8.5.1.27.3	Diagram.....	320
8.5.1.27.4	Fields.....	321
8.5.1.28	Value of OTP Bank1 Word3 (Memory Related Info.) (HW_OCOTP_MEM3).....	321
8.5.1.28.1	Offset.....	321
8.5.1.28.2	Function.....	321
8.5.1.28.3	Diagram.....	321
8.5.1.28.4	Fields.....	322
8.5.1.29	Value of OTP Bank1 Word4 (Memory Related Info.) (HW_OCOTP_MEM4).....	322
8.5.1.29.1	Offset.....	322
8.5.1.29.2	Function.....	322
8.5.1.29.3	Diagram.....	322
8.5.1.29.4	Fields.....	323
8.5.1.30	Value of OTP Bank1 Word5 (Analog Info.) (HW_OCOTP_ANA0).....	323
8.5.1.30.1	Offset.....	323
8.5.1.30.2	Function.....	323
8.5.1.30.3	Diagram.....	323
8.5.1.30.4	Fields.....	324
8.5.1.31	Value of OTP Bank1 Word6 (Analog Info.) (HW_OCOTP_ANA1).....	324
8.5.1.31.1	Offset.....	324
8.5.1.31.2	Function.....	324
8.5.1.31.3	Diagram.....	324
8.5.1.31.4	Fields.....	325
8.5.1.32	Value of OTP Bank1 Word7 (Analog Info.) (HW_OCOTP_ANA2).....	325

Section number	Title	Page
	8.5.1.32.1 Offset.....	325
	8.5.1.32.2 Function.....	325
	8.5.1.32.3 Diagram.....	325
	8.5.1.32.4 Fields.....	326
8.5.1.33	Shadow Register for OTP Bank3 Word0 (SRK Hash) (HW_OCOTP_SRK0).....	326
	8.5.1.33.1 Offset.....	326
	8.5.1.33.2 Function.....	326
	8.5.1.33.3 Diagram.....	326
	8.5.1.33.4 Fields.....	327
8.5.1.34	Shadow Register for OTP Bank3 Word1 (SRK Hash) (HW_OCOTP_SRK1).....	327
	8.5.1.34.1 Offset.....	327
	8.5.1.34.2 Function.....	327
	8.5.1.34.3 Diagram.....	327
	8.5.1.34.4 Fields.....	328
8.5.1.35	Shadow Register for OTP Bank3 Word2 (SRK Hash) (HW_OCOTP_SRK2).....	328
	8.5.1.35.1 Offset.....	328
	8.5.1.35.2 Function.....	328
	8.5.1.35.3 Diagram.....	328
	8.5.1.35.4 Fields.....	329
8.5.1.36	Shadow Register for OTP Bank3 Word3 (SRK Hash) (HW_OCOTP_SRK3).....	329
	8.5.1.36.1 Offset.....	329
	8.5.1.36.2 Function.....	329
	8.5.1.36.3 Diagram.....	329
	8.5.1.36.4 Fields.....	330
8.5.1.37	Shadow Register for OTP Bank3 Word4 (SRK Hash) (HW_OCOTP_SRK4).....	330
	8.5.1.37.1 Offset.....	330
	8.5.1.37.2 Function.....	330
	8.5.1.37.3 Diagram.....	330
	8.5.1.37.4 Fields.....	331

Section number	Title	Page
8.5.1.38	Shadow Register for OTP Bank3 Word5 (SRK Hash) (HW_OCOTP_SRK5).....	331
8.5.1.38.1	Offset.....	331
8.5.1.38.2	Function.....	331
8.5.1.38.3	Diagram.....	331
8.5.1.38.4	Fields.....	332
8.5.1.39	Shadow Register for OTP Bank3 Word6 (SRK Hash) (HW_OCOTP_SRK6).....	332
8.5.1.39.1	Offset.....	332
8.5.1.39.2	Function.....	332
8.5.1.39.3	Diagram.....	332
8.5.1.39.4	Fields.....	333
8.5.1.40	Shadow Register for OTP Bank3 Word7 (SRK Hash) (HW_OCOTP_SRK7).....	333
8.5.1.40.1	Offset.....	333
8.5.1.40.2	Function.....	333
8.5.1.40.3	Diagram.....	333
8.5.1.40.4	Fields.....	334
8.5.1.41	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP0).....	334
8.5.1.41.1	Offset.....	334
8.5.1.41.2	Function.....	334
8.5.1.41.3	Diagram.....	334
8.5.1.41.4	Fields.....	335
8.5.1.42	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP1).....	335
8.5.1.42.1	Offset.....	335
8.5.1.42.2	Function.....	335
8.5.1.42.3	Diagram.....	335
8.5.1.42.4	Fields.....	336
8.5.1.43	Value of OTP Bank4 Word2 (MAC Address) (HW_OCOTP_MAC0).....	336
8.5.1.43.1	Offset.....	336
8.5.1.43.2	Function.....	336
8.5.1.43.3	Diagram.....	336

Section number	Title	Page
	8.5.1.43.4 Fields.....	337
8.5.1.44	Value of OTP Bank4 Word3 (MAC Address) (HW_OCOTP_MAC1).....	337
	8.5.1.44.1 Offset.....	337
	8.5.1.44.2 Function.....	337
	8.5.1.44.3 Diagram.....	337
	8.5.1.44.4 Fields.....	338
8.5.1.45	Value of OTP Bank4 Word4 (MAC Address) (HW_OCOTP_GP3).....	338
	8.5.1.45.1 Offset.....	338
	8.5.1.45.2 Function.....	338
	8.5.1.45.3 Diagram.....	338
	8.5.1.45.4 Fields.....	339
8.5.1.46	Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (HW_OCOTP_GP1)...	339
	8.5.1.46.1 Offset.....	339
	8.5.1.46.2 Function.....	339
	8.5.1.46.3 Diagram.....	339
	8.5.1.46.4 Fields.....	340
8.5.1.47	Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (HW_OCOTP_GP2)...	340
	8.5.1.47.1 Offset.....	340
	8.5.1.47.2 Function.....	340
	8.5.1.47.3 Diagram.....	340
	8.5.1.47.4 Fields.....	341
8.5.1.48	Value of OTP Bank5 Word0 (SW GP1) (HW_OCOTP_SW_GP1).....	341
	8.5.1.48.1 Offset.....	341
	8.5.1.48.2 Function.....	341
	8.5.1.48.3 Diagram.....	341
	8.5.1.48.4 Fields.....	342
8.5.1.49	Value of OTP Bank5 Word1 (SW GP2) (HW_OCOTP_SW_GP20).....	342
	8.5.1.49.1 Offset.....	342
	8.5.1.49.2 Function.....	342

Section number	Title	Page
8.5.1.49.3	Diagram.....	342
8.5.1.49.4	Fields.....	343
8.5.1.50	Value of OTP Bank5 Word2 (SW GP2) (HW_OCOTP_SW_GP21).....	343
8.5.1.50.1	Offset.....	343
8.5.1.50.2	Function.....	343
8.5.1.50.3	Diagram.....	343
8.5.1.50.4	Fields.....	344
8.5.1.51	Value of OTP Bank5 Word3 (SW GP2) (HW_OCOTP_SW_GP22).....	344
8.5.1.51.1	Offset.....	344
8.5.1.51.2	Function.....	344
8.5.1.51.3	Diagram.....	344
8.5.1.51.4	Fields.....	345
8.5.1.52	Value of OTP Bank5 Word4 (SW GP2) (HW_OCOTP_SW_GP23).....	345
8.5.1.52.1	Offset.....	345
8.5.1.52.2	Function.....	345
8.5.1.52.3	Diagram.....	345
8.5.1.52.4	Fields.....	346
8.5.1.53	Value of OTP Bank5 Word5 (Misc Conf) (HW_OCOTP_MISC_CONF0).....	346
8.5.1.53.1	Offset.....	346
8.5.1.53.2	Function.....	346
8.5.1.53.3	Diagram.....	346
8.5.1.53.4	Fields.....	347
8.5.1.54	Value of OTP Bank5 Word6 (Misc Conf) (HW_OCOTP_MISC_CONF1).....	347
8.5.1.54.1	Offset.....	347
8.5.1.54.2	Function.....	347
8.5.1.54.3	Diagram.....	347
8.5.1.54.4	Fields.....	348
8.5.1.55	Value of OTP Bank5 Word7 (SRK Revoke) (HW_OCOTP_SRK_REVOKE).....	348
8.5.1.55.1	Offset.....	348

Section number	Title	Page
8.5.1.55.2	Function.....	348
8.5.1.55.3	Diagram.....	348
8.5.1.55.4	Fields.....	349

Chapter 9 True Random Number Generator (TRNG)

9.1	Chip-specific TRNG information.....	351
9.2	Standalone True Random Number Generator (SA-TRNG).....	351
9.2.1	Standalone True Random Number Generator Block Diagram.....	352
9.2.2	TRNG Functional Description.....	352
9.2.3	SA-TRNG hardware functional description.....	353
9.2.3.1	Software Use Cases for the Stand Alone TRNG.....	353
9.2.4	Another TRNG usage example.....	355
9.2.5	TRNG register descriptions.....	355
9.2.5.1	TRNG Memory map.....	355
9.2.5.2	Miscellaneous Control Register (MCTL).....	357
9.2.5.2.1	Offset.....	357
9.2.5.2.2	Function.....	357
9.2.5.2.3	Diagram.....	357
9.2.5.2.4	Fields.....	358
9.2.5.3	Statistical Check Miscellaneous Register (SCMISC).....	359
9.2.5.3.1	Offset.....	359
9.2.5.3.2	Function.....	360
9.2.5.3.3	Diagram.....	360
9.2.5.3.4	Fields.....	360
9.2.5.4	Poker Range Register (PKRRNG).....	360
9.2.5.4.1	Offset.....	360
9.2.5.4.2	Function.....	361
9.2.5.4.3	Diagram.....	361
9.2.5.4.4	Fields.....	361

Section number	Title	Page
9.2.5.5	Poker Square Calculation Result Register (PKRSQ).....	361
9.2.5.5.1	Offset.....	361
9.2.5.5.2	Function.....	362
9.2.5.5.3	Diagram.....	362
9.2.5.5.4	Fields.....	362
9.2.5.6	Poker Maximum Limit Register (PKRMAX).....	362
9.2.5.6.1	Offset.....	363
9.2.5.6.2	Function.....	363
9.2.5.6.3	Diagram.....	363
9.2.5.6.4	Fields.....	363
9.2.5.7	Seed Control Register (SDCTL).....	364
9.2.5.7.1	Offset.....	364
9.2.5.7.2	Function.....	364
9.2.5.7.3	Diagram.....	364
9.2.5.7.4	Fields.....	364
9.2.5.8	Sparse Bit Limit Register (SBLIM).....	364
9.2.5.8.1	Offset.....	365
9.2.5.8.2	Function.....	365
9.2.5.8.3	Diagram.....	365
9.2.5.8.4	Fields.....	365
9.2.5.9	Total Samples Register (TOTSAM).....	366
9.2.5.9.1	Offset.....	366
9.2.5.9.2	Function.....	366
9.2.5.9.3	Diagram.....	366
9.2.5.9.4	Fields.....	366
9.2.5.10	Frequency Count Minimum Limit Register (FRQMIN).....	367
9.2.5.10.1	Offset.....	367
9.2.5.10.2	Function.....	367
9.2.5.10.3	Diagram.....	367

Section number	Title	Page
	9.2.5.10.4 Fields.....	367
9.2.5.11	Frequency Count Maximum Limit Register (FRQMAX).....	367
	9.2.5.11.1 Offset.....	368
	9.2.5.11.2 Function.....	368
	9.2.5.11.3 Diagram.....	368
	9.2.5.11.4 Fields.....	368
9.2.5.12	Frequency Count Register (FRQCNT).....	368
	9.2.5.12.1 Offset.....	369
	9.2.5.12.2 Function.....	369
	9.2.5.12.3 Diagram.....	369
	9.2.5.12.4 Fields.....	369
9.2.5.13	Statistical Check Monobit Count Register (SCMC).....	370
	9.2.5.13.1 Offset.....	370
	9.2.5.13.2 Function.....	370
	9.2.5.13.3 Diagram.....	370
	9.2.5.13.4 Fields.....	370
9.2.5.14	Statistical Check Monobit Limit Register (SCML).....	370
	9.2.5.14.1 Offset.....	371
	9.2.5.14.2 Function.....	371
	9.2.5.14.3 Diagram.....	371
	9.2.5.14.4 Fields.....	371
9.2.5.15	Statistical Check Run Length 1 Limit Register (SCR1L).....	372
	9.2.5.15.1 Offset.....	372
	9.2.5.15.2 Function.....	372
	9.2.5.15.3 Diagram.....	372
	9.2.5.15.4 Fields.....	372
9.2.5.16	Statistical Check Run Length 1 Count Register (SCR1C).....	373
	9.2.5.16.1 Offset.....	373
	9.2.5.16.2 Function.....	373

Section number	Title	Page
9.2.5.16.3	Diagram.....	373
9.2.5.16.4	Fields.....	374
9.2.5.17	Statistical Check Run Length 2 Count Register (SCR2C).....	374
9.2.5.17.1	Offset.....	374
9.2.5.17.2	Function.....	374
9.2.5.17.3	Diagram.....	375
9.2.5.17.4	Fields.....	375
9.2.5.18	Statistical Check Run Length 2 Limit Register (SCR2L).....	375
9.2.5.18.1	Offset.....	376
9.2.5.18.2	Function.....	376
9.2.5.18.3	Diagram.....	376
9.2.5.18.4	Fields.....	376
9.2.5.19	Statistical Check Run Length 3 Limit Register (SCR3L).....	377
9.2.5.19.1	Offset.....	377
9.2.5.19.2	Function.....	377
9.2.5.19.3	Diagram.....	377
9.2.5.19.4	Fields.....	378
9.2.5.20	Statistical Check Run Length 3 Count Register (SCR3C).....	378
9.2.5.20.1	Offset.....	378
9.2.5.20.2	Function.....	378
9.2.5.20.3	Diagram.....	379
9.2.5.20.4	Fields.....	379
9.2.5.21	Statistical Check Run Length 4 Count Register (SCR4C).....	379
9.2.5.21.1	Offset.....	379
9.2.5.21.2	Function.....	379
9.2.5.21.3	Diagram.....	380
9.2.5.21.4	Fields.....	380
9.2.5.22	Statistical Check Run Length 4 Limit Register (SCR4L).....	380
9.2.5.22.1	Offset.....	380

Section number	Title	Page
9.2.5.22.2	Function.....	381
9.2.5.22.3	Diagram.....	381
9.2.5.22.4	Fields.....	381
9.2.5.23	Statistical Check Run Length 5 Count Register (SCR5C).....	382
9.2.5.23.1	Offset.....	382
9.2.5.23.2	Function.....	382
9.2.5.23.3	Diagram.....	382
9.2.5.23.4	Fields.....	382
9.2.5.24	Statistical Check Run Length 5 Limit Register (SCR5L).....	383
9.2.5.24.1	Offset.....	383
9.2.5.24.2	Function.....	383
9.2.5.24.3	Diagram.....	383
9.2.5.24.4	Fields.....	384
9.2.5.25	Statistical Check Run Length 6+ Count Register (SCR6PC).....	384
9.2.5.25.1	Offset.....	384
9.2.5.25.2	Function.....	384
9.2.5.25.3	Diagram.....	385
9.2.5.25.4	Fields.....	385
9.2.5.26	Statistical Check Run Length 6+ Limit Register (SCR6PL).....	385
9.2.5.26.1	Offset.....	385
9.2.5.26.2	Function.....	386
9.2.5.26.3	Diagram.....	386
9.2.5.26.4	Fields.....	386
9.2.5.27	Status Register (STATUS).....	386
9.2.5.27.1	Offset.....	387
9.2.5.27.2	Function.....	387
9.2.5.27.3	Diagram.....	387
9.2.5.27.4	Fields.....	388
9.2.5.28	Entropy Read Register (ENT0 - ENT15).....	389

Section number	Title	Page
	9.2.5.28.1 Offset.....	389
	9.2.5.28.2 Function.....	389
	9.2.5.28.3 Diagram.....	389
	9.2.5.28.4 Fields.....	389
9.2.5.29	Statistical Check Poker Count 1 and 0 Register (PKRCNT10).....	390
	9.2.5.29.1 Offset.....	390
	9.2.5.29.2 Function.....	390
	9.2.5.29.3 Diagram.....	390
	9.2.5.29.4 Fields.....	390
9.2.5.30	Statistical Check Poker Count 3 and 2 Register (PKRCNT32).....	391
	9.2.5.30.1 Offset.....	391
	9.2.5.30.2 Function.....	391
	9.2.5.30.3 Diagram.....	391
	9.2.5.30.4 Fields.....	391
9.2.5.31	Statistical Check Poker Count 5 and 4 Register (PKRCNT54).....	392
	9.2.5.31.1 Offset.....	392
	9.2.5.31.2 Function.....	392
	9.2.5.31.3 Diagram.....	392
	9.2.5.31.4 Fields.....	392
9.2.5.32	Statistical Check Poker Count 7 and 6 Register (PKRCNT76).....	393
	9.2.5.32.1 Offset.....	393
	9.2.5.32.2 Function.....	393
	9.2.5.32.3 Diagram.....	393
	9.2.5.32.4 Fields.....	393
9.2.5.33	Statistical Check Poker Count 9 and 8 Register (PKRCNT98).....	394
	9.2.5.33.1 Offset.....	394
	9.2.5.33.2 Function.....	394
	9.2.5.33.3 Diagram.....	394
	9.2.5.33.4 Fields.....	394

Section number	Title	Page
9.2.5.34	Statistical Check Poker Count B and A Register (PKRCNTBA).....	395
9.2.5.34.1	Offset.....	395
9.2.5.34.2	Function.....	395
9.2.5.34.3	Diagram.....	395
9.2.5.34.4	Fields.....	395
9.2.5.35	Statistical Check Poker Count D and C Register (PKRCNTDC).....	396
9.2.5.35.1	Offset.....	396
9.2.5.35.2	Function.....	396
9.2.5.35.3	Diagram.....	396
9.2.5.35.4	Fields.....	396
9.2.5.36	Statistical Check Poker Count F and E Register (PKRCNTFE).....	397
9.2.5.36.1	Offset.....	397
9.2.5.36.2	Function.....	397
9.2.5.36.3	Diagram.....	397
9.2.5.36.4	Fields.....	397
9.2.5.37	Security Configuration Register (SEC_CFG).....	398
9.2.5.37.1	Offset.....	398
9.2.5.37.2	Function.....	398
9.2.5.37.3	Diagram.....	398
9.2.5.37.4	Fields.....	398
9.2.5.38	Interrupt Control Register (INT_CTRL).....	399
9.2.5.38.1	Offset.....	399
9.2.5.38.2	Function.....	399
9.2.5.38.3	Diagram.....	399
9.2.5.38.4	Fields.....	400
9.2.5.39	Mask Register (INT_MASK).....	400
9.2.5.39.1	Offset.....	400
9.2.5.39.2	Function.....	400
9.2.5.39.3	Diagram.....	401

Section number	Title	Page
	9.2.5.39.4 Fields.....	401
9.2.5.40	Interrupt Status Register (INT_STATUS).....	401
	9.2.5.40.1 Offset.....	402
	9.2.5.40.2 Function.....	402
	9.2.5.40.3 Diagram.....	402
	9.2.5.40.4 Fields.....	402
9.2.5.41	Version ID Register (MS) (VID1).....	403
	9.2.5.41.1 Offset.....	403
	9.2.5.41.2 Function.....	403
	9.2.5.41.3 Diagram.....	403
	9.2.5.41.4 Fields.....	404
9.2.5.42	Version ID Register (LS) (VID2).....	404
	9.2.5.42.1 Offset.....	404
	9.2.5.42.2 Function.....	404
	9.2.5.42.3 Diagram.....	404
	9.2.5.42.4 Fields.....	404

Chapter 10 Secure Non-Volatile Storage (SNVS)

10.1	Chip-specific SNVS information.....	407
10.2	SNVS introduction.....	407
10.2.1	SNVS feature list.....	408
10.2.2	SNVS functional description.....	410
10.3	SNVS Structure.....	411
10.3.1	SNVS power domains.....	413
10.3.2	SNVS clock sources.....	414
10.4	Security violation policy.....	414
10.4.1	Security state machine.....	415
10.4.2	SNVS interrupts, alarms, and security violations.....	419
10.4.3	Configuring SNVS's response to a security violation or tamper event.....	419

Section number	Title	Page
10.4.4	Tamper detects.....	421
10.4.5	SNVS_LP security violation/tamper policy.....	422
10.4.6	Power glitch detector (PGD).....	422
10.4.7	High Assurance Counter.....	423
10.5	Runtime Procedures.....	424
10.5.1	Using SNVS Timer Facilities.....	424
10.5.1.1	SNVS_HP Real Time Counter.....	424
10.5.1.2	SNVS_LP Secure Real Time Counter (SRTC).....	425
10.5.1.3	RTC/SRTC control bits setting.....	426
10.5.1.4	Reading RTC and SRTC values.....	427
10.5.2	Using Other SNVS Registers.....	427
10.5.2.1	Using the General-Purpose Register.....	428
10.5.2.2	Using the Monotonic Counter (MC).....	428
10.6	Configuring Master Key checking and control.....	429
10.6.1	Error Code for the OTPMK.....	431
10.6.2	Provisioning the Zeroizable Master Key.....	431
10.7	Reset and Initialization of SNVS.....	433
10.7.1	Power glitch detector (PGD).....	435
10.7.2	SNVS_LP Secure Real Time Counter (SRTC).....	436
10.7.3	Initialization Checklists.....	437
10.7.3.1	Checklist for Initialization of the SNVS HP.....	438
10.7.3.2	Checklist for Initialization of the SNVS LP.....	439
10.8	SNVS register descriptions.....	440
10.8.1	SNVS Memory map.....	441
10.8.2	SNVS_HP Lock Register (HPLR).....	442
10.8.2.1	Offset.....	442
10.8.2.2	Function.....	443
10.8.2.3	Diagram.....	443
10.8.2.4	Fields.....	443

Section number	Title	Page
10.8.3	SNVS_HP Command Register (HPCOMR).....	445
10.8.3.1	Offset.....	445
10.8.3.2	Function.....	445
10.8.3.3	Diagram.....	446
10.8.3.4	Fields.....	446
10.8.4	SNVS_HP Control Register (HPCR).....	449
10.8.4.1	Offset.....	449
10.8.4.2	Function.....	449
10.8.4.3	Diagram.....	449
10.8.4.4	Fields.....	449
10.8.5	SNVS_HP Security Interrupt Control Register (HPSICR).....	451
10.8.5.1	Offset.....	451
10.8.5.2	Function.....	452
10.8.5.3	Diagram.....	452
10.8.5.4	Fields.....	452
10.8.6	SNVS_HP Security Violation Control Register (HPSVCR).....	453
10.8.6.1	Offset.....	453
10.8.6.2	Function.....	453
10.8.6.3	Diagram.....	453
10.8.6.4	Fields.....	454
10.8.7	SNVS_HP Status Register (HPSR).....	455
10.8.7.1	Offset.....	455
10.8.7.2	Function.....	455
10.8.7.3	Diagram.....	455
10.8.7.4	Fields.....	456
10.8.8	SNVS_HP Security Violation Status Register (HPSVSR).....	458
10.8.8.1	Offset.....	458
10.8.8.2	Function.....	458
10.8.8.3	Diagram.....	458

Section number	Title	Page
10.8.8.4	Fields.....	459
10.8.9	SNVS_HP High Assurance Counter IV Register (HPHACIVR).....	460
10.8.9.1	Offset.....	460
10.8.9.2	Function.....	460
10.8.9.3	Diagram.....	461
10.8.9.4	Fields.....	461
10.8.10	SNVS_HP High Assurance Counter Register (HPHACR).....	461
10.8.10.1	Offset.....	461
10.8.10.2	Function.....	461
10.8.10.3	Diagram.....	462
10.8.10.4	Fields.....	462
10.8.11	SNVS_HP Real Time Counter MSB Register (HPRTCMR).....	462
10.8.11.1	Offset.....	462
10.8.11.2	Function.....	463
10.8.11.3	Diagram.....	463
10.8.11.4	Fields.....	463
10.8.12	SNVS_HP Real Time Counter LSB Register (HPRTCLR).....	463
10.8.12.1	Offset.....	463
10.8.12.2	Function.....	464
10.8.12.3	Diagram.....	464
10.8.12.4	Fields.....	464
10.8.13	SNVS_HP Time Alarm MSB Register (HPTAMR).....	464
10.8.13.1	Offset.....	464
10.8.13.2	Function.....	464
10.8.13.3	Diagram.....	465
10.8.13.4	Fields.....	465
10.8.14	SNVS_HP Time Alarm LSB Register (HPTALR).....	465
10.8.14.1	Offset.....	465
10.8.14.2	Function.....	465

Section number	Title	Page
10.8.14.3	Diagram.....	466
10.8.14.4	Fields.....	466
10.8.15	SNVS_LP Lock Register (LPLR).....	466
10.8.15.1	Offset.....	466
10.8.15.2	Function.....	466
10.8.15.3	Diagram.....	467
10.8.15.4	Fields.....	467
10.8.16	SNVS_LP Control Register (LPCR).....	469
10.8.16.1	Offset.....	469
10.8.16.2	Function.....	469
10.8.16.3	Diagram.....	469
10.8.16.4	Fields.....	469
10.8.17	SNVS_LP Master Key Control Register (LPMKCR).....	472
10.8.17.1	Offset.....	472
10.8.17.2	Function.....	472
10.8.17.3	Diagram.....	472
10.8.17.4	Fields.....	473
10.8.18	SNVS_LP Security Violation Control Register (LPSVCR).....	473
10.8.18.1	Offset.....	474
10.8.18.2	Function.....	474
10.8.18.3	Diagram.....	474
10.8.18.4	Fields.....	474
10.8.19	SNVS_LP Tamper Detectors Configuration Register (LPTDCR).....	475
10.8.19.1	Offset.....	475
10.8.19.2	Function.....	475
10.8.19.3	Diagram.....	476
10.8.19.4	Fields.....	476
10.8.20	SNVS_LP Status Register (LPSR).....	477
10.8.20.1	Offset.....	478

Section number	Title	Page
10.8.20.2	Function.....	478
10.8.20.3	Diagram.....	478
10.8.20.4	Fields.....	478
10.8.21	SNVS_LP Secure Real Time Counter MSB Register (LPSRTCMR).....	480
10.8.21.1	Offset.....	480
10.8.21.2	Function.....	480
10.8.21.3	Diagram.....	480
10.8.21.4	Fields.....	480
10.8.22	SNVS_LP Secure Real Time Counter LSB Register (LPSRTCLR).....	481
10.8.22.1	Offset.....	481
10.8.22.2	Function.....	481
10.8.22.3	Diagram.....	481
10.8.22.4	Fields.....	482
10.8.23	SNVS_LP Time Alarm Register (LPTAR).....	482
10.8.23.1	Offset.....	482
10.8.23.2	Function.....	482
10.8.23.3	Diagram.....	482
10.8.23.4	Fields.....	482
10.8.24	SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR).....	483
10.8.24.1	Offset.....	483
10.8.24.2	Function.....	483
10.8.24.3	Diagram.....	483
10.8.24.4	Fields.....	483
10.8.25	SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR).....	484
10.8.25.1	Offset.....	484
10.8.25.2	Function.....	484
10.8.25.3	Diagram.....	484
10.8.25.4	Fields.....	485
10.8.26	SNVS_LP Power Glitch Detector Register (LPPGDR).....	485

Section number	Title	Page
10.8.26.1	Offset.....	485
10.8.26.2	Function.....	485
10.8.26.3	Diagram.....	486
10.8.26.4	Fields.....	486
10.8.27	SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias).....	486
10.8.27.1	Offset.....	486
10.8.27.2	Function.....	486
10.8.27.3	Diagram.....	487
10.8.27.4	Fields.....	487
10.8.28	SNVS_LP Zeroizable Master Key Register (LPZMKR0 - LPZMKR7).....	487
10.8.28.1	Offset.....	487
10.8.28.2	Function.....	487
10.8.28.3	Diagram.....	488
10.8.28.4	Fields.....	488
10.8.29	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_alias - LPGPR3_alias).....	488
10.8.29.1	Offset.....	488
10.8.29.2	Function.....	489
10.8.29.3	Diagram.....	489
10.8.29.4	Fields.....	489
10.8.30	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0 - LPGPR3).....	490
10.8.30.1	Offset.....	490
10.8.30.2	Function.....	490
10.8.30.3	Diagram.....	490
10.8.30.4	Fields.....	490
10.8.31	SNVS_HP Version ID Register 1 (HPVIDR1).....	491
10.8.31.1	Offset.....	491
10.8.31.2	Function.....	491
10.8.31.3	Diagram.....	491
10.8.31.4	Fields.....	491

Section number	Title	Page
10.8.32	SNVS_HP Version ID Register 2 (HPVIDR2).....	492
10.8.32.1	Offset.....	492
10.8.32.2	Function.....	492
10.8.32.3	Diagram.....	492
10.8.32.4	Fields.....	493

Chapter 11 System JTAG Controller (SJC)

11.1	Overview.....	495
11.1.1	Features.....	495
11.1.2	Modes of Operation.....	495
11.2	External Signals.....	497
11.2.1	External Signal Overview.....	498
11.2.2	TAP Controller.....	498
11.2.3	Accessing ExtraDebug Registers.....	500
11.3	Boundary Scan Register (BSR)	502
11.4	SoC JTAG Instruction Register (SJIR)	502
11.4.1	ID_CODE Instruction (IDCODE)	503
11.4.2	SAMPLE/PRELOAD Instruction	504
11.4.3	EXTEST Instruction.....	505
11.4.4	HIGHZ Instruction.....	505
11.4.5	BYPASS Instruction	505
11.4.6	ENABLE_ExtraDebug Instruction	506
11.4.7	ENTER_DEBUG instruction	506
11.4.8	EXTEST_PULSE instruction	507
11.4.9	EXTEST_TRAIN instruction.....	507
11.5	Security.....	507
11.5.1	JTAG Security Modes	508
11.5.1.1	Mode 1: No Debug - Maximum Security.....	508
11.5.1.2	Mode 2: Secure JTAG - High Security	508

Section number	Title	Page
11.5.1.2.1	Challenge/Response Mechanism in System JTAG Mode.....	509
11.5.1.3	Mode 3: JTAG Enabled - Low Security	510
11.5.2	Software Enabled JTAG.....	510
11.5.3	Kill Trace.....	511
11.5.4	SJC Disable Fuse	512
11.6	Functional Description.....	513
11.6.1	Static Core Debug.....	513
11.6.2	Reset Mechanism.....	513
11.7	Initialization/Application Information.....	514
11.8	SJC Memory Map/Register Definition.....	514
11.8.1	General Purpose Unsecured Status Register 1 (SJC_GPUSR1).....	516
11.8.2	General Purpose Unsecured Status Register 2 (SJC_GPUSR2).....	518
11.8.3	General Purpose Unsecured Status Register 3 (SJC_GPUSR3).....	519
11.8.4	General Purpose Secured Status Register (SJC_GPSSR).....	519
11.8.5	Debug Control Register (SJC_DCR).....	520
11.8.6	Security Status Register (SJC_SSR).....	522
11.8.7	General Purpose Clocks Control Register (SJC_GPCCR).....	525

Chapter 1

Security Overview

1.1 Chapter overview

This chapter provides an overview of the following chip security components, explaining the purpose and features of each of them.

- High Assurance Boot (HAB) feature in the system boot
- Secure Non-Volatile Storage (SNVS) with the security monitor, key storage, and real-time clock
- Data co-processor (DCP) with cryptographic acceleration
- Hardware encryption and hash algorithm engine for AES128 and SHA-1/256
- True Random Number Generator (TRNG)
- On-chip One-Time Programmable Element Controller (OCOTP_CTRL) with on-chip electrical fuse arrays
- Central Security Unit (CSU)
- System JTAG Controller (SJC) with secure debug
- Bus Encryption Engine (BEE) for on-the-fly FlexSPI(QSPI) Flash decryption

1.2 Feature summary

This figure shows a simplified diagram of the security subsystem:

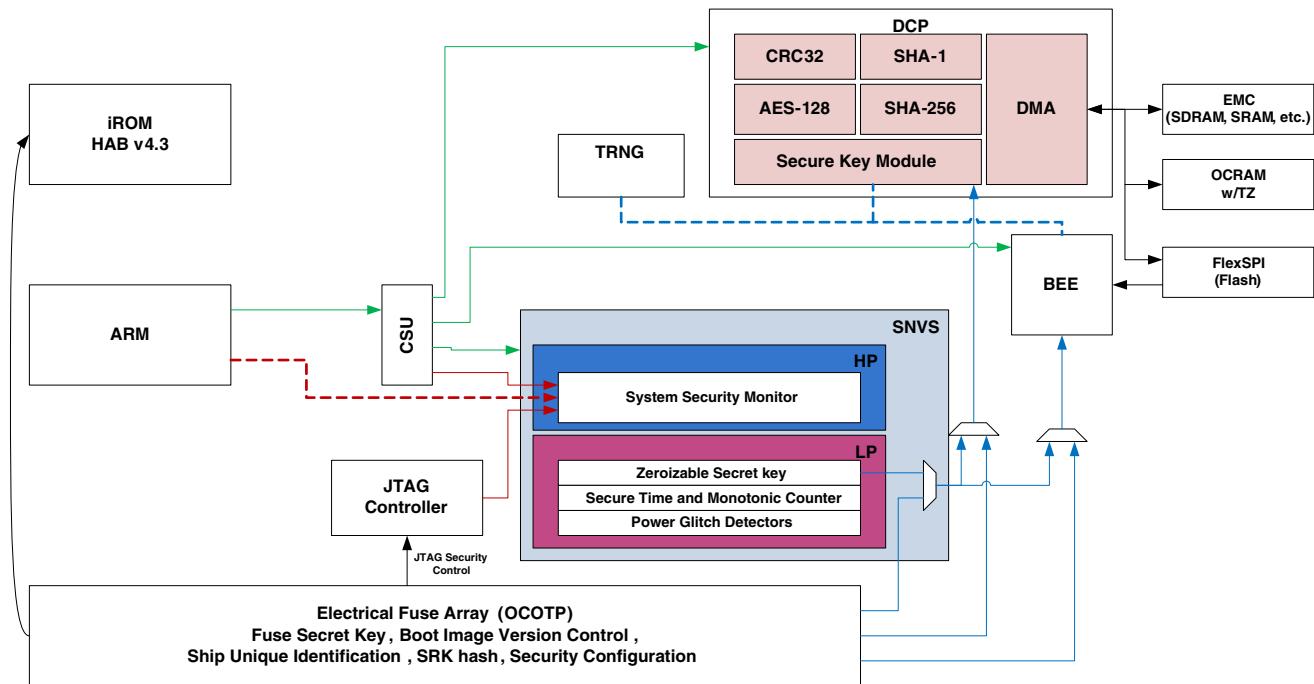


Figure 1-1. Security subsystem (simplified)

This diagram represents an example of the CSU use.

All platforms built using this chip share a general need for security, though the specific security requirements vary greatly from platform to platform. For example, portable consumer devices need to protect a different type and cost of assets than the automotive or industrial platforms. Each market must be protected against different kinds of attacks. The platform designers need an appropriate set of counter measures to meet the security needs of their specific platform.

To help the platform designers to meet the requirements of each market, the chip incorporates a range of security features. Most of these features provide protection against specific kinds of attack, and can be configured for different levels according to the required degree of protection. These features are designed to work together or independently. They can be also integrated with the appropriate software to create defensive layers. In addition, the chip includes a general-purpose accelerator that enhances the performance of selected industry-standard cryptographic algorithms.

The security features include:

- Secure High-Assurance Boot
 - Security library embedded in the tamper-proof on-chip ROM
 - Authenticated boot, which protects against unauthorized software
 - Verification of the code signature during boot
 - RSA-1024/2048/3072/4096 keys anchored to the OTP fingerprint (SHA-256)

- Encrypted boot which protects the software confidentiality
- Runs every time the chip is reset
- Image version control/image revocation (on-chip OTP-based)
- Secure storage
 - Off-chip storage protection using AES-128 and the chip's unique hardware-only key
- Hardware cryptographic accelerators
 - Symmetric: AES-128,
 - Hash message digest: SHA-1, SHA-256,
- True and pseudorandom number generator
- On-chip secure real-time clock with autonomous power domain
- Secure debugging
 - Configurable protection against unauthorized JTAG manipulation
 - Three security levels + a complete JTAG disable
 - Support for JTAG port secure reopening for field return debugging
- Universal unique ID
- Electrical fuses (OTP Memory)
- Hardware bus encryption
 - AES-128 encryption, supporting ECB and CTR modes
 - Non-secured access filtering

1.3 High-Assurance Boot (HAB)

The HAB, which is the high-assurance boot feature in the system boot ROM, detects and prevents the execution of unauthorized software (malware) during the boot sequence.

When the unauthorized software is permitted to gain control of the boot sequence, it can be used for a variety of goals, such as exposing stored secrets; circumventing access controls to sensitive data, services, or networks, or for repurposing the platform. The unauthorized software can enter the platform during upgrades or reprovisioning, or when booting from the USB connections or removable devices.

The HAB protects against unauthorized software by:

- Using digital signatures to recognize the authentic software. This enables you to boot the device to a known initial state and run the software signed by the device manufacturer.
- Using code encryption to protect the confidential software during off-chip storage. When activated, the HAB decrypts the software loaded into the RAM before execution.

1.3.1 HAB process flow

The following figure shows the flow for creating and verifying digital signatures. The top half of this figure shows the signing process, which is performed off-chip. The bottom half shows the verification process performed on-chip during every system boot.

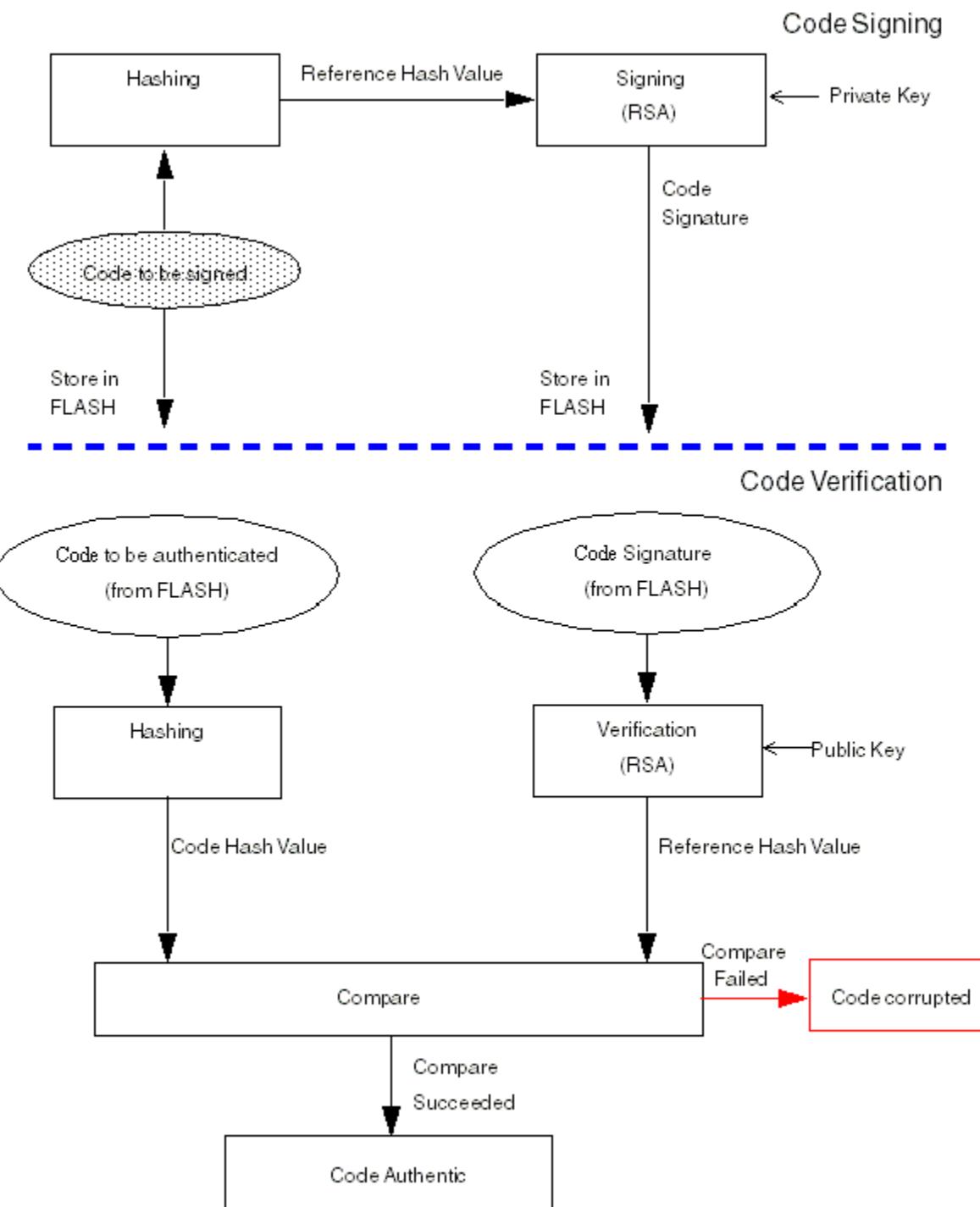


Figure 1-2. Code signing and authentication processes

The original software is programmed into the flash memory (or any other boot device) along with the signature. The HAB uses a public key to recover the reference hash value from the signature; it then compares the reference hash value to the current hash value

calculated from the software in the flash. If the contents of the flash are modified either intentionally or unintentionally, the two hash values do not match and the verification fails.

1.3.2 HAB feature summary

The HAB features:

- Enforced internal boot via on-chip masked ROM
- Authentication of software loaded from any boot device (including the USB download)
- Authenticated decryption of software loaded from any boot device (including USB download) using the AES keys (128-bit)
- CMS PKCS#1 signature verification using the RSA public keys (from 1024-bit to 4096-bit) and the SHA-256 hash algorithm
- Public Key Infrastructure (PKI) support using X.509v3 certificates
- Root public key fingerprint in the manufacturer-programmable on-chip fuses
- Multiple root public keys with revocation by fuses
- Initialization of other security components
- Authenticated USB download fall-over on any security failure
- Open configuration for development purposes and non-secure platforms
- Closed configuration for shipping secure platforms

On the chip, the HAB is integrated with these security features:

- The HAB initializes the SNVS security monitor state machine. A successful secure boot with the HAB is required for the platform software to gain access to use the DCP master secret key selected by SNVS.
- The HAB reads the root public key fingerprint, revocation mask, and security configuration from the OCOTP_CTRL.
- The HAB initializes the CSU.
- The HAB can use the DCP to accelerate hash calculations.

1.4 Secure Non-Volatile Storage (SNVS) module

- Provides a non-volatile real-time clock maintained by a coin-cell battery during system power down for use in both the secure and non-secure platforms.
- Protects the secure real-time clock against rollback attacks in time-sensitive protocols such as DRM and PKI
- Deters replay attacks in time-independent protocols such as certificate or firmware revocations

- Handles tamper detection and tamper reaction to defend sensitive data and operations against compromise, both at run-time and during system power-down
- Controls the access to the OTP master secret key used by the DCP to protect confidential data in the off-chip storage
- Provides non-volatile highly protected storage for an alternative master secret key

1.4.1 SNVS architecture

The SNVS is partitioned into two sections: a low-power part (SNVS_LP) and a high-power part (SNVS_HP).

The SNVS_LP block is in the always-powered-up domain. It is isolated from the rest of the logic by isolation cells which are library-instantiated cells that insure that the powered-up logic is not corrupted when the power goes down in the rest of the chip.

The SNVS_LP has these functional units:

- Zeroizable Master Key
- Secure non-rollover real-time counter with alarm
- Non-rollover monotonic counter
- Power glitch detector
- General-purpose register
- Control and status registers

The SNVS_HP is in the chip power-supply domain. The SNVS_HP provides an interface between the SNVS_LP and the rest of the system. The access to the SNVS_LP registers can be gained through the SNVS_HP only when it is powered up according to the access permission policy.

The SNVS_HP has these functional units:

- IP bus interface
- SNVS_LP interface
- System Security Monitor (SSM)
- Zeroizable Master Key programming mechanism
- Master Key control block
- Non-secure real-time counter with alarm
- Control and status registers

1.5 Data Co-Processor (DCP)

For security purposes, the Data Co-Processor (DCP) provides hardware acceleration for the cryptographic algorithms. The features of DCP include:

- Encryption Algorithms: AES-128 (ECB and CBC modes)
- Hashing Algorithms: SHA-1 and SHA-256
- CRC-32
- Key selection from the SNVS, DCP internal key storage, or general memory
- Internal Memory for storing up to four AES-128 keys—when a key is written to a key slot it can be read only by the DCP AES-128 engine
- IP slave interface
- DMA

1.6 Standalone True Random Number Generator (TRNG)

The SA-TRNG is hardware accelerator module that generates a 512-bit entropy as needed by an entropy consuming module or by other post processing functions. A typical entropy consumer is a pseudo random number generator (PRNG) which can be implemented to achieve both true randomness and cryptographic strength random numbers using the SA-TRNG output as its entropy seed. The PRNG is not part of this module.

1.7 OCOTP_CTRL

The OCOTP_CTRL provides the primary user-visible mechanism for interfacing with the on-chip fuses. These fuses' uses include:

- Unique chip identifiers
- Mask revision numbers
- Cryptographic keys
- Security configuration
- Boot characteristics
- Various control signals requiring permanent non-volatility

For security purposes, the fuses protect the confidentiality or integrity of the critical security data against both the software attacks and the board-level hardware attacks.

The OCOTP_CTRL provides:

- Shadow cache of fuse values, loaded at reset, before the system boot
- Ability to read and override the fuse values in the shadow cache (does not affect the fuse element)
- Ability to read the fuses directly (ignoring the shadow cache)
- Ability to write (program) the fuses by software or JTAG
- Fuses and shadow cache bits enforce read-protect, override-protect, and write-protect
- Lock fuses for selected fuse fields
- Scan protection
- Volatile software-accessible signals which can be used for software control of hardware elements (not requiring non-volatility).

1.8 Central Security Unit (CSU)

Central Security Unit (CSU) sets access control policies between the bus masters and bus slaves, enabling the peripherals to be separated into distinct security domains. This protects against the indirect unauthorized access to data which occurs when the software programs a DMA bus master to access addresses that the software itself is prohibited from accessing directly. Configuring the DMA bus master privileges in the CSU consistently with the software privileges defends against such indirect unauthorized access.

The CSU provides:

- Configuration of peripheral access permissions for peripherals that are unable to control their own access permissions
- Configuration of bus master privileges for bus masters that are unable to control their own privileges
- Optional locking of the individual CSU settings until the next power-on reset

1.9 System JTAG Controller (SJC)

The JTAG port provides debug access to hardware blocks, including the ARM processor and the system bus. This enables program control and manipulation as well as visibility to the chip peripherals and memory.

The JTAG port must be accessible during initial platform development, manufacturing tests, and general troubleshooting. Given its capabilities, JTAG manipulation is a known attack vector for accessing sensitive data and gaining control over software execution.

Bus Encryption Engine (BEE)

The System JTAG Controller (SJC) protects against the whole range of attacks based on unauthorized JTAG manipulation. It also provides a JTAG port that conforms to the IEEE 1149.1 and IEEE 1149.6 (AC) standards for BSR (boundary-scan) testing.

The SJC provides these security levels:

- The JTAG Disabled-JTAG use is permanently blocked.
- The No-Debug-All security sensitive JTAG features are permanently blocked.
- The Secure JTAG-JTAG use is restricted (as in the No-Debug level) unless a secret-key challenge/response protocol is successfully executed.
- The JTAG Enabled-JTAG use is unrestricted.

The security levels are selected via the e-fuse configuration.

1.9.1 Scan protection

The chip includes further scan protection logic for those SJC modes where the JTAG use is allowed. This ensures that the access to critical security values is protected as follows:

- The chip is reset when entering the scan mode.
- All modules are reset two clock cycles before receiving the scan-enable indication.
- The chip cannot exit the scan mode without a reset.
- The security modules (including SNVS, CSU, and OCOTP_CTRL) have an additional scan-protection logic to protect the sensitive internal data and functionality.

See the "System JTAG Controller (SJC)" chapter in the Security Reference Manual for more information on the SJC.

1.10 Bus Encryption Engine (BEE)

The Bus Encryption Engine (BEE) is implemented as an on-the-fly decryption engine, which is used for decrypting cypher context of FlexSPI. The main features of the BEE are:

- Standard AXI interconnection
- On-the-fly AES-128 decryption, supporting ECB and CTR modes
- Aliased memory space support. Address remapping for up to two individual regions
- Independent AES Key management for those two individual regions
- Bus access pattern optimization with the aid of the local store and forward buffer

- Non-secured access filtering based on the security label of the access
- Illegal access check and filtering

Chapter 2

Security System Integration

2.1 Master ID allocation

This table summarizes the master IDs for all system master modules:

Table 2-1. Master IDs

Module	Master ID
Core Platform	000b
eDMA	001b
DCP	010b
All others	011b

2.2 System-level SNVS connections

2.2.1 SNVS clock tamper input

The system provides an automatic detection of the external SRTC clock state and provides an alternative internal clock source when a failure is detected. When the external clock detection fails, the system automatically switches to an internally generated clock from a ring oscillator on the processor. When the clock on the RTC_xtali appears again, the system switches back to the external clock. Thus, the clock tamper detection function is reduced to a detection of a power failure in the SNVS power domain or the SNVS reset.

The clock tamper detects these scenarios (thus generating tamper):

- No clock detection

Security access error

- Under-clock tamper—when the clock (32 KHz) goes below the programmed low-threshold value
- Over-clock tamper—when the clock goes above the programmed high-threshold value

2.2.2 System security violation alarm signals monitored by SNVS

The SNVS supports several system security violation alarm inputs, as shown in the following table. This allocation is related to the HPSVCR and LPSVCR SNVS registers.

Table 2-2. SNVS system security violation alarm input signals

SNVS registers			Source	Description
HPSVSR register bit field	HPSVCR register bit field	LPSVCR register bit field		
SEC_VIO1	SV_CFG1	SV_EN1	SJC	JTAG active
SEC_VIO2	SV_CFG2	SV_EN2	WDOG2	Watchdog 2 reset
SEC_VIO3	SV_CFG3	SV_EN3	(Reserved)	—
SEC_VIO4	SV_CFG4	SV_EN4	SRC	Internal boot

2.3 Security access error

The system slave modules can be configured to return a bus-access error when a security-violating access is detected, using the SEC_ERR_RESP bit (GPR10[2] register):

- When set, the slave modules return a bus-error indication on a non-proper security level access.
- When cleared, the operation does not proceed on a non-proper security level access, but the slave modules do not indicate an error.

The bit is set by default, enabling the error indications. The SEC_RRR_RESP itself can be locked, preventing further modifications by the LOCK_SEC_ERR_RESP bit (GPR10[18]) to assure the system security integrity.

For more information, see the general registers in the chip reference manual.

2.4 OCRAM TrustZone support

The OCRAM supports TrustZone and non-TrustZone accesses to the internal on-chip RAM. There is an option to configure a TrustZone-only access region.

When the OCRAM_TZSECURE_REGION[SECURE_ENBL] bit in the OCRAM module is set, the STARTADDR and ENDADDR bit fields in this register establish the region of OCRAM that can only be accessed (both read and write) in accordance with the execution mode policy defined in the "Execution Mode Access Policy" section of the CSU chapter. If this bit is cleared, the entire OCRAM can be accessed in either the secure or non-secure modes. The secure-region addresses and the SECURE_ENBL are programmed through the IOMUXC and should be modified when no transactions are occurring on the OCRAM bus.

The TrustZone bits are described in the "Programmable Registers" section of the chip reference manual.

NOTE

The ENDADDR bit field is not configurable. Its value is the last address of the OCRAM space. The STARTADDR granularity is 4 KB.

This figure shows the OCRAM schematic connectivity:

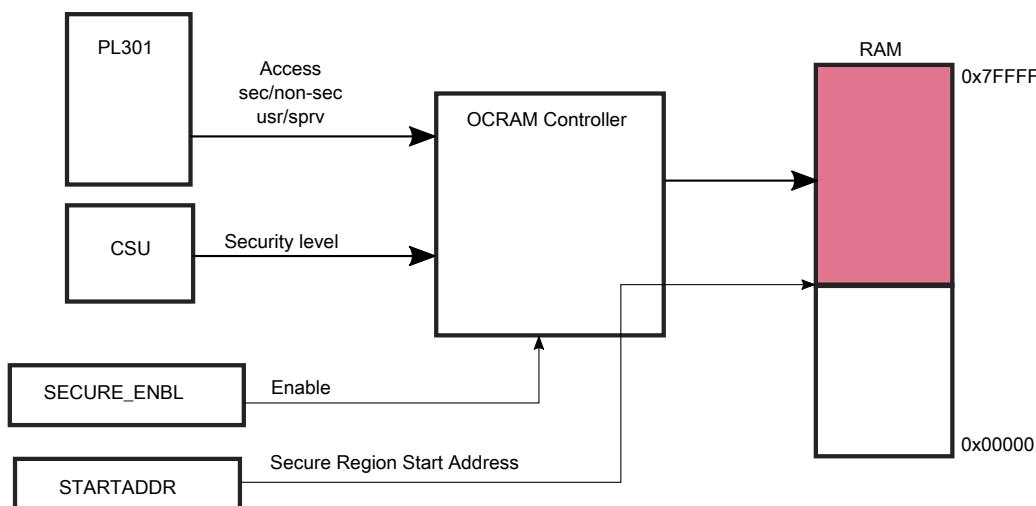


Figure 2-1. OCRAM schematic connectivity

2.5 Watchdog mechanism

The chip has two WDOG modules: WDOG1 and WDOG2 (TZ). Both modules are disabled by default after the reset. The WDOG1 is configured during the boot. The WDOG2 is dedicated for the Secure-World purposes and is only activated by the TrustZone software (if required).

The WDOG module operates as follows:

- If servicing does not take place, the timer times out and asserts the internal system reset signal (wdog_rst_B) which goes to the SRC (System Reset Controller).
- The interrupt can be generated before the timer actually times out.
- The wdog_rst_B signal can be activated by software.
- There is a power-down counter that is enabled out of any reset. This counter has a fixed time-out period of 16 seconds after which it asserts the WDOG_B signal.

2.6 Security configuration

The following figure illustrates the typical device security-configuration lifecycle which starts from the IC fabrication and continues with the OEM development and assembly through to the final product in the end user's hands. It also shows the option for field-return debugging and re-testing at either the OEM or NXP facilities. Note that the field-return configuration is required for NXP to run test patterns even on non-secure products (open configuration).

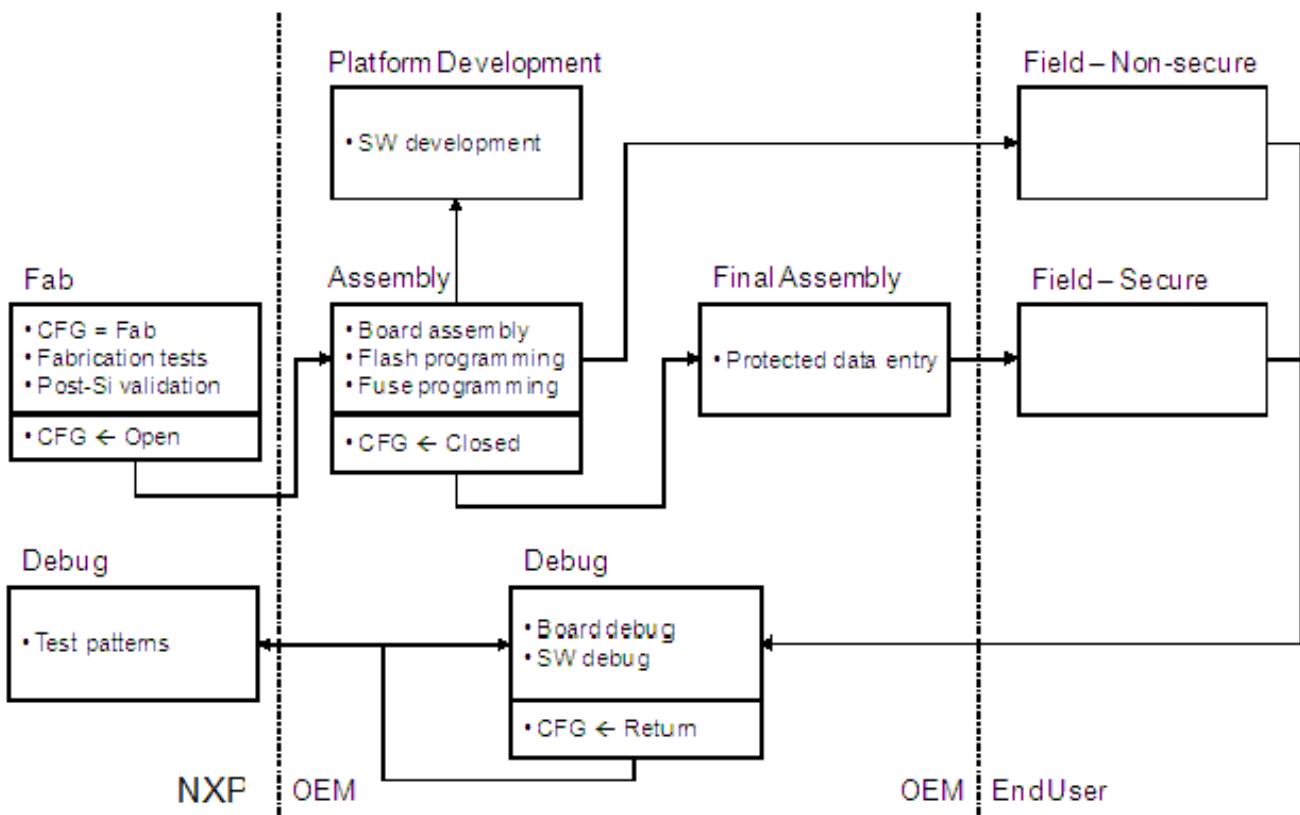


Figure 2-2. Device security configuration life cycle

2.7 Field return for retest procedure

Manufacturers can enable the debugging restrictions designed to protect the device keys and other sensitive data on the devices shipped to the end users. These debugging restrictions include these measures:

- Disabling NXP test modes (using the DIR_BT_DIS fuse).
- Disabling JTAG (using the JTAG_SMODE or SJC_DISABLE fuses).
- Preventing the execution of unauthorized bootloader software (using the SEC_CONFIG[1] fuse).

Naturally, these debugging restrictions also constrain the legitimate debugging of the field-return devices with suspected faults. The chip includes a field-return configuration to enable the legitimate debugging, including the possibility for NXP to run the test modes on returned parts. The field-return configuration:

- Enables NXP test modes (overriding the DIR_BT_DIS fuse).
- Enables JTAG (overriding the JTAG_SMODE and SJC_DISABLE fuses).
- Enables the execution of unsigned bootloader software as in the open configuration (overriding the SEC_CONFIG[1] fuse).

To protect the sensitive data already provisioned, the field-return configuration permanently disables the access to the device keys (including access from the DCP modules).

The entry to the field-return configuration is strictly controlled to deter inadvertent, unauthorized, or widespread use.

- The FIELD_RETURN fuse is protected by the FIELD_RETURN_LOCK sticky bit in the OCOTP_CTRL fuse controller.
- Before leaving the boot ROM, the FIELD_RETURN_LOCK bit is set by default, so that the FIELD_RETURN fuse cannot be burned.
- Setting the FIELD_RETURN_LOCK bit can be avoided by including the unlock command in the CSF or DCD (open configuration only) which provides:
 - The CSF and bootloader software pass signature verification (Closed configuration).
 - The unlock command arguments match the value in the UNIQUE_ID fuses on the device.

Field return for retest procedure

The typical mass-production bootloader on the shipped devices has no unlock command, so the entry to the field-return configuration requires booting to a special bootloader which is customized for a single device. For the closed devices, a special bootloader must be signed for that single device, so that it cannot be used to unlock other devices even if it leaks to the end users.

The boot flow to activate the field-return configuration is shown in this figure:

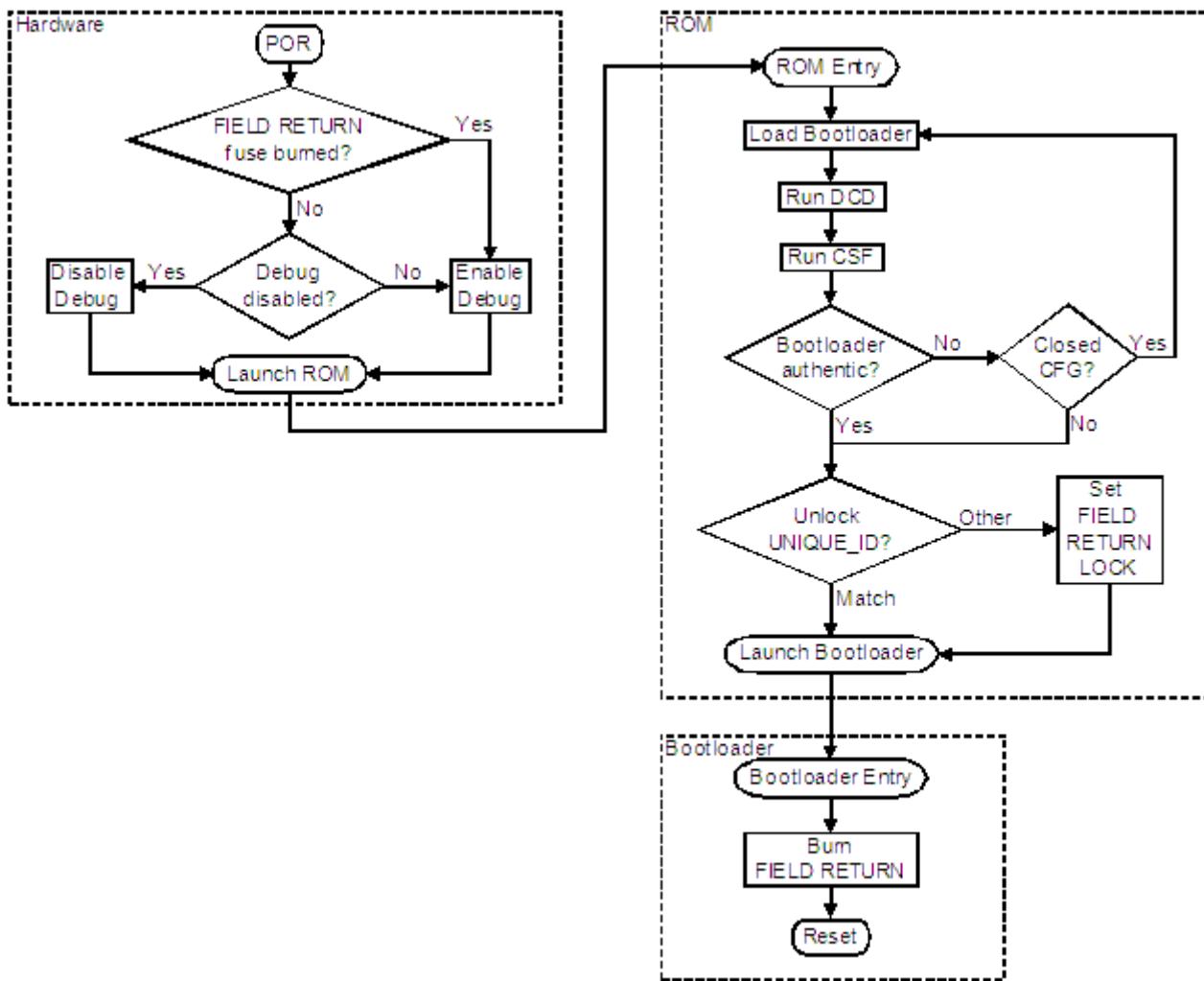


Figure 2-3. HAB FIELD_RETURN flowchart

When the FIELD_RETURN fuse is burned, the part is nearly returned to its open state after the next POR, including:

- Reverting back the value of the DIR_BT_DIS fuse to enable the NXP test mode to run.
- Enabling the JTAG (overriding blocking by other means).
- Enabling the unsigned images to execute, as in the open configuration.

- Block access to the sensitive keys provided by the OCOTP_CTRL or SNVS to .
- Note that the field-return configuration is required for NXP to run the test patterns even on non-secure products (open configuration).

Chapter 3

System Boot

3.1 Chip-specific Boot Information

This device has various peripherals supported by the ROM bootloader.

Table 3-1. ROM Bootloader Peripheral PinMux

Peripheral	Instance	Port (IO function)	PAD	Mode
LPUART	1	LPUART1_TX	GPIO_AD_B0_12	ALT2
		LPUART1_RX	GPIO_AD_B0_13	ALT2
LPSPI	1	LPSPI1_SCK	GPIO_SD_B0_00	ALT4
		LPSPI1_SDO	GPIO_SD_B0_02	ALT4
		LPSPI1_SDI	GPIO_SD_B0_03	ALT4
		LPSPI1_PCS0	GPIO_SD_B0_01	ALT4
	2	LPSPI2_SCK	GPIO_SD_B1_07	ALT4
		LPSPI2_SDO	GPIO_SD_B1_08	ALT4
		LPSPI2_SDI	GPIO_SD_B1_09	ALT4
		LPSPI2_PCS0	GPIO_SD_B1_06	ALT4
	3	LPSPI3_SCK	GPIO_AD_B0_00	ALT7
		LPSPI3_SDO	GPIO_AD_B0_01	ALT7
		LPSPI3_SDI	GPIO_AD_B0_02	ALT7
		LPSPI3_PCS0	GPIO_AD_B0_03	ALT7
	4	LPSPI4_SCK	GPIO_B0_03	ALT3
		LPSPI4_SDO	GPIO_B0_02	ALT3
		LPSPI4_SDI	GPIO_B0_01	ALT3
		LPSPI4_PCS0	GPIO_B0_00	ALT3
SEMC NAND	-	SEMC_DATA00	GPIO_EMC_00	ALT0
		SEMC_DATA01	GPIO_EMC_01	ALT0
		SEMC_DATA02	GPIO_EMC_02	ALT0
		SEMC_DATA03	GPIO_EMC_03	ALT0
		SEMC_DATA04	GPIO_EMC_04	ALT0
		SEMC_DATA05	GPIO_EMC_05	ALT0

Table continues on the next page...

Table 3-1. ROM Bootloader Peripheral PinMux (continued)

		SEMC_DATA06	GPIO_EMC_06	ALT0
		SEMC_DATA07	GPIO_EMC_07	ALT0
		SEMC_DATA08	GPIO_EMC_30	ALT0
		SEMC_DATA09	GPIO_EMC_31	ALT0
		SEMC_DATA10	GPIO_EMC_32	ALT0
		SEMC_DATA11	GPIO_EMC_33	ALT0
		SEMC_DATA12	GPIO_EMC_34	ALT0
		SEMC_DATA13	GPIO_EMC_35	ALT0
		SEMC_DATA14	GPIO_EMC_36	ALT0
		SEMC_DATA15	GPIO_EMC_37	ALT0
		SEMC_ADDR9	GPIO_EMC_18	ALT0
		SEMC_ADDR11	GPIO_EMC_19	ALT0
		SEMC_ADDR12	GPIO_EMC_20	ALT0
		SEMC_BA1	GPIO_EMC_22	ALT0
		SEMC_CSX0	GPIO_EMC_41	ALT0
SEMC NOR	-	SEMC_DATA00	GPIO_EMC_00	ALT0
		SEMC_DATA01	GPIO_EMC_01	ALT0
		SEMC_DATA02	GPIO_EMC_02	ALT0
		SEMC_DATA03	GPIO_EMC_03	ALT0
		SEMC_DATA04	GPIO_EMC_04	ALT0
		SEMC_DATA05	GPIO_EMC_05	ALT0
		SEMC_DATA06	GPIO_EMC_06	ALT0
		SEMC_DATA07	GPIO_EMC_07	ALT0
		SEMC_DATA08	GPIO_EMC_30	ALT0
		SEMC_DATA09	GPIO_EMC_31	ALT0
		SEMC_DATA10	GPIO_EMC_32	ALT0
		SEMC_DATA11	GPIO_EMC_33	ALT0
		SEMC_DATA12	GPIO_EMC_34	ALT0
		SEMC_DATA13	GPIO_EMC_35	ALT0
		SEMC_DATA14	GPIO_EMC_36	ALT0
		SEMC_DATA15	GPIO_EMC_37	ALT0
		SEMC_ADDR0	GPIO_EMC_09	ALT0
		SEMC_ADDR1	GPIO_EMC_10	ALT0
		SEMC_ADDR2	GPIO_EMC_11	ALT0
		SEMC_ADDR3	GPIO_EMC_12	ALT0
		SEMC_ADDR4	GPIO_EMC_13	ALT0
		SEMC_ADDR5	GPIO_EMC_14	ALT0
		SEMC_ADDR6	GPIO_EMC_15	ALT0
		SEMC_ADDR7	GPIO_EMC_16	ALT0
		SEMC_ADDR11	GPIO_EMC_19	ALT0

Table continues on the next page...

Table 3-1. ROM Bootloader Peripheral PinMux (continued)

		SEMC_ADDR12	GPIO_EMC_20	ALT0
		SEMC_BA0	GPIO_EMC_21	ALT0
		SEMC_BA1	GPIO_EMC_22	ALT0
		SEMC_CSX0	GPIO_EMC_41	ALT0
SD	1	USDHC1_CMD	GPIO_SD_B0_00	ALT0
		USDHC1_CLK	GPIO_SD_B0_01	ALT0
		USDHC1_DATA0	GPIO_SD_B0_02	ALT0
		USDHC1_DATA1	GPIO_SD_B0_03	ALT0
		USDHC1_DATA2	GPIO_SD_B0_04	ALT0
		USDHC1_DATA3	GPIO_SD_B0_05	ALT0
		USDHC1_CD_B	GPIO_SD_B1_12	ALT6
	2	USDHC2_DATA3	GPIO_SD_B1_00	ALT0
		USDHC2_DATA2	GPIO_SD_B1_01	ALT0
		USDHC2_DATA1	GPIO_SD_B1_02	ALT0
		USDHC2_DATA0	GPIO_SD_B1_03	ALT0
		USDHC2_CLK	GPIO_SD_B1_04	ALT0
		USDHC2_CMD	GPIO_SD_B1_05	ALT0
		USDHC2_RESET_B	GPIO_SD_B1_06	ALT0
FlexSPI	1	USDHC2_DATA4	GPIO_SD_B1_08	ALT0
		USDHC2_DATA5	GPIO_SD_B1_09	ALT0
		USDHC2_DATA6	GPIO_SD_B1_10	ALT0
		USDHC2_DATA7	GPIO_SD_B1_11	ALT0
		FLEXSPI_B_DATA3	GPIO_SD_B1_00	ALT1
		FLEXSPI_B_DATA2	GPIO_SD_B1_01	ALT1
		FLEXSPI_B_DATA1	GPIO_SD_B1_02	ALT1
		FLEXSPI_B_DATA0	GPIO_SD_B1_03	ALT1
		FLEXSPI_B_SCLK	GPIO_SD_B1_04	ALT1
		FLEXSPI_B_DQS	GPIO_SD_B0_05	ALT4
		FLEXSPI_B_SS0_B	GPIO_SD_B0_04	ALT4
		FLEXSPI_B_SS1_B	GPIO_SD_B0_01	ALT6
		FLEXSPI_A_DQS	GPIO_SD_B1_05	ALT1
		FLEXSPI_A_SS0_B	GPIO_SD_B1_06	ALT1
		FLEXSPI_A_SS1_B	GPIO_SD_B0_00	ALT6
		FLEXSPI_A_SCLK	GPIO_SD_B1_07	ALT1
		FLEXSPI_A_DATA0	GPIO_SD_B1_08	ALT1
		FLEXSPI_A_DATA1	GPIO_SD_B1_09	ALT1
		FLEXSPI_A_DATA2	GPIO_SD_B1_10	ALT1
		FLEXSPI_A_DATA3	GPIO_SD_B1_11	ALT1
		FLEXSPI_A_DATA7	GPIO_SD_B1_00	ALT1
		FLEXSPI_A_DATA6	GPIO_SD_B1_01	ALT1

Table continues on the next page...

Table 3-1. ROM Bootloader Peripheral PinMux (continued)

		FLEXSPI_A_DATA5	GPIO_SD_B1_02	ALT1
		FLEXSPI_A_DATA4	GPIO_SD_B1_03	ALT1
FlexSPI (secondary pinmux option)	1	FLEXSPI_A_SS0_B	GPIO_AD_B1_15	ALT0
		FLEXSPI_A_SCLK	GPIO_AD_B1_14	ALT0
		FLEXSPI_A_DQS	GPIO_AD_B1_09	ALT0
		FLEXSPI_A_DATA0	GPIO_AD_B1_13	ALT0
		FLEXSPI_A_DATA1	GPIO_AD_B1_12	ALT0
		FLEXSPI_A_DATA2	GPIO_AD_B1_11	ALT0
		FLEXSPI_A_DATA3	GPIO_AD_B1_10	ALT0

3.2 Overview

The boot process begins at the Power-On Reset (POR) where the hardware reset logic forces the ARM core to begin the execution starting from the on-chip boot ROM.

The boot ROM code uses the state of the internal register BOOT_MODE[1:0] as well as the state of various eFUSES and/or GPIO settings to determine the boot flow behavior of the device.

The main features of the ROM include:

- Support for booting from various boot devices
- Serial downloader support (USB OTG and UART)
- Device Configuration Data (DCD) and plugin
- Digital signature and encryption based High-Assurance Boot (HAB)
- Wake-up from the low-power modes
- Encrypted XIP on Serial NOR via FlexSPI interface powered by BEE and DCP controller

The boot ROM supports these boot devices:

- Serial NOR Flash via FlexSPI
- Serial NAND Flash via FlexSPI
- Parallel NOR Flash via SEMC
- RAWNAND Flash via SEMC
- SD/MMC
- SPI NOR/EEPROM via LPSPI

The boot ROM uses the state of the BOOT_MODE and eFUSES to determine the boot device. For development purposes, the eFUSES used to determine the boot device may be overridden using the GPIO pin inputs.

The boot ROM code also allows to download the programs to be run on the device. The example is a provisioning program that can make further use of the serial connection to provide a boot device with a new image. Typically, the provisioning program is downloaded to the internal RAM and allows to program the boot devices, such as the SD/MMC flash. The ROM serial downloader uses a high-speed USB in a non-stream mode connection.

The boot ROM allows waking up from the low-power modes. On reset, the ROM checks the power gating status register. When waking from the low-power mode, the core skips loading an image from the boot device and jumps to the address saved in PERSISTENT_ENTRY0.

The Device Configuration Data (DCD) feature allows the boot ROM code to obtain the SOC configuration data from an external program image residing on the boot device. As an example, the DCD can be used to program the SDRAM controller for optimal settings, improving the boot performance. The DCD is restricted to the memory areas and peripheral addresses that are considered essential for the boot purposes (see [Write data command](#)).

A key feature of the boot ROM is the ability to perform a secure boot, also known as a High-Assurance Boot (HAB). This is supported by the HAB security library which is a subcomponent of the ROM code. The HAB uses a combination of hardware and software together with the Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized programs. Before the HAB allows the user image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as a part of the final program image. If configured to do so, the ROM verifies the signatures using the public keys included in the program image. In addition to supporting the digital signature verification to authenticate the program images, the encrypted boot is also supported. The encrypted boot can be used to prevent the cloning of the program image directly off the boot device. A secure boot with HAB can be performed on all boot devices supported on the chip in addition to the serial downloader. The HAB library in the boot ROM also provides the API functions, allowing the additional boot chain components (bootloaders) to extend the secure boot chain. The out-of-fab setting for the SEC_CONFIG is the open configuration, in which the ROM/HAB performs the image authentication, but all authentication errors are ignored and the image is still allowed to execute.

3.3 Boot modes

During reset, the chip checks the power gating controller status register.

Boot modes

During boot, the core's behavior is defined by the boot mode pin settings, as described in [Boot mode pin settings](#). When waking up from the low-power boot mode, the core skips the clock settings. The boot ROM checks that the PERSISTENT_ENTRY0 (see [Persistent bits](#)) is a pointer to a valid address space (OCRAM, or DDR). If the PERSISTENT_ENTRY0 is a pointer to a valid range, it starts the execution using the entry point from the PERSISTENT_ENTRY0 register. If the PERSISTENT_ENTRY0 is a pointer to an invalid range, the core performs the system reset.

3.3.1 Boot mode pin settings

The device has four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT_MODE register.

The BOOT_MODE is initialized by sampling the BOOT_MODE0 and BOOT_MODE1 inputs on the rising edge of the POR_B. After these inputs are sampled, their subsequent state does not affect the contents of the BOOT_MODE internal register. The state of the internal BOOT_MODE register may be read from the BMOD[1:0] field of the SRC Boot Mode Register (SRC_SBMR2). The available boot modes are: Boot From Fuses, serial boot via USB, and Internal Boot. See this table for settings:

Table 3-2. Boot MODE pin settings

BOOT_MODE[1:0]	Boot Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

3.3.2 High-level boot sequence

The figure found here show the high-level boot ROM code flow.

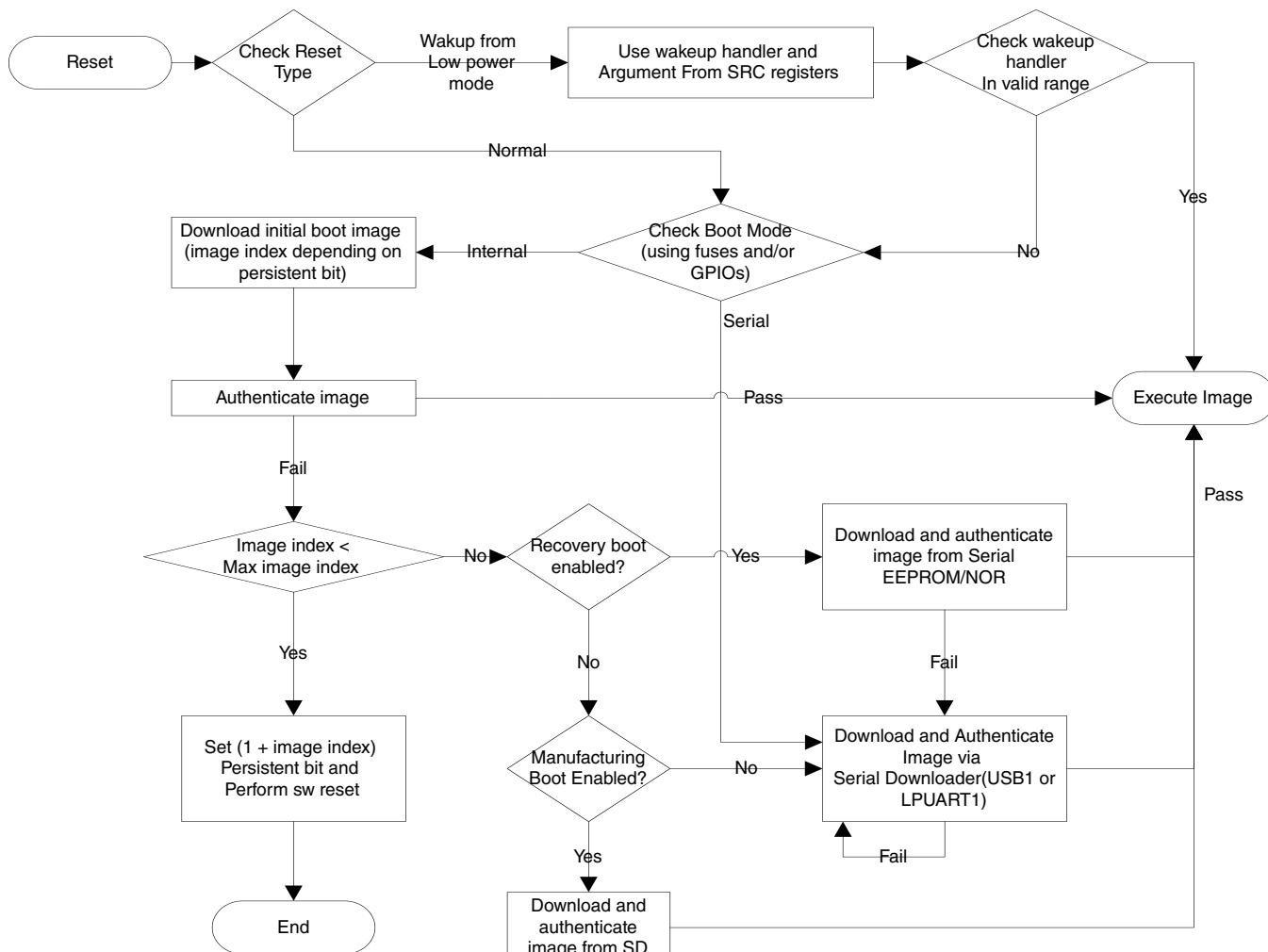


Figure 3-1. Boot flow

3.3.3 Boot From Fuses mode (BOOT_MODE[1:0] = 00b)

A value of 00b in the BOOT_MODE[1:0] register selects the Boot From Fuses mode.

This mode is similar to the Internal Boot mode described in [Internal Boot mode \(BOOT_MODE\[1:0\] = 0b10\)](#) with one difference. In this mode, the GPIO boot override pins are ignored. The boot ROM code uses the boot eFUSE settings only. This mode also supports a secure boot using HAB.

If set to Boot From Fuses, the boot flow is controlled by the BT_FUSE_SEL eFUSE value. If BT_FUSE_SEL = 0, indicating that the boot device (for example, flash, SD/MMC) was not programmed yet, the boot flow jumps directly to the Serial Downloader. If BT_FUSE_SEL = 1, the normal boot flow is followed, where the ROM attempts to boot from the selected boot device.

The first time a board is used, the default eFUSES may be configured incorrectly for the hardware on the platform. In such case, the Boot ROM code may try to boot from a device that does not exist. This may cause an electrical/logic violation on some pads. Using the Boot From Fuses mode addresses this problem.

Setting the BT_FUSE_SEL=0 forces the ROM code to jump directly to the Serial Downloader. This allows a bootloader to be downloaded which can then provision the boot device with a program image and blow the BT_FUSE_SEL and the other boot configuration eFUSES. After the reset, the boot ROM code determines that the BT_FUSE_SEL is blown (BT_FUSE_SEL = 1) and the ROM code performs an internal boot according to the new eFUSE settings. This allows the user to set BOOT_MODE[1:0]=00b on a production device and burn the fuses on the same device (by forcing the entry to the Serial Downloader), without changing the value of the BOOT_MODE[1:0] or the pullups/pulldowns on the BOOT_MODE pins.

3.3.4 Serial Downloader (BOOT_MODE[1:0] = 01b)

The Serial Downloader provides a means to download a Program Image to the chip over USB or UART serial connection. In this mode, typically a host PC can communicate to the ROM bootloader using serial download protocol. Serial downloader and the protocol are discussed in [Serial Downloader](#).

3.3.5 Internal Boot mode (BOOT_MODE[1:0] = 0b10)

A value of 0b10 in the BOOT_MODE[1:0] register selects the Internal Boot mode. In this mode, the processor continues to execute the boot code from the internal boot ROM.

The boot code performs the hardware initialization, loads the program image from the chosen boot device, performs the image validation using the HAB library (see [Boot security settings](#)), and then jumps to an address derived from the program image. If an error occurs during the internal boot, the boot code jumps to the Serial Downloader (see [Serial Downloader \(BOOT_MODE\[1:0\] = 01b\)](#)). A secure boot using the HAB is possible in all the three boot modes.

When set to the Internal Boot, the boot flow may be controlled by a combination of eFUSE settings with an option of overriding the fuse settings using the General Purpose I/O (GPIO) pins. The GPIO Boot Select FUSE (BT_FUSE_SEL) determines whether the ROM uses the GPIO pins for a selected number of configuration parameters or eFUSES in this mode.

- If BT_FUSE_SEL = 1, all boot options are controlled by the eFUSES described in [Boot eFUSE descriptions](#).
- If BT_FUSE_SEL = 0, the specific boot configuration parameters may be set using the GPIO pins rather than eFUSES. The fuses that can be overridden when in this mode are indicated in the GPIO column of [Boot eFUSE descriptions](#). [GPIO boot overrides](#) provides the details of the GPIO pins.

The use of the GPIO overrides is intended for development since these pads are used for other purposes in the deployed products. NXP recommends controlling the boot configuration by the eFUSES in the deployed products and reserving the use of the GPIO mode for the development and testing purposes only.

3.3.6 Boot security settings

The internal boot modes use one of three security configurations.

- Closed: This level is intended for use with shipping-secure products. All HAB functions are executed and the security hardware is initialized (the Security Controller or SNVS enters the Secure state), the DCD is processed if present, and the program image is authenticated by the HAB before its execution. All detected errors are logged, and the boot flow is aborted with the control being passed to the serial downloader. At this level, the execution does not leave the internal ROM unless the target executable image is authenticated.
- Open: This level is intended for use in non-secure products or during the development phases of a secure product. All HAB functions are executed as for a closed device. The security hardware is initialized (except for the SNVS which is left in the Non-Secure state), the DCD is processed if present, and the program image is authenticated by the HAB before its execution. All detected errors are logged, but have no influence on the boot flow which continues as if the errors did not occur. This configuration is useful for a secure product development because the program image runs even if the authentication data is missing or incorrect, and the error log can be examined to determine the cause of the authentication failure.
- Field Return: This level is intended for the parts returned from the shipped products.

NOTE

If the DIR_BT_DIS eFuse is not blown, the authentication may be bypassed. In this case the system is not secure.

3.4 Device configuration

This section describes the external inputs that control the behavior of the Boot ROM code.

This includes the boot device selection (FlexSPI NOR, FlexSPI NAND, Parallel NOR, SD, MMC, and so on), boot device configuration (SD bus width, speed, and so on), and other. In general, the source for this configuration comes from the eFUSES embedded inside the chip. However, certain configuration parameters can be sourced from the GPIO pins, allowing further flexibility during the development process.

3.4.1 Boot eFUSE descriptions

This table is a comprehensive list of the configuration parameters that the ROM uses.

Table 3-3. Boot eFUSE descriptions

Fuse	Config uratio n	Definition	GPIO ¹	Shipped value	Settings ²
DIR_BT_DIS	OEM	Disables the NXP reserved modes. Must be set for the secure boot. .	NA	0	0—The reserved NXP modes are enabled. 1—The reserved NXP modes are disabled.
BT_FUSE_SEL	OEM	In the Internal Boot mode BOOT_MODE[1:0] = 10, the BT_FUSE_SEL fuse determines whether the boot settings indicated by a Yes in the GPIO column are controlled by the GPIO pins or the eFUSE settings in the On-Chip OTP Controller (OCOTP). In the Boot From Fuse mode BOOT_MODE[1:0] = 00, the BT_FUSE_SEL fuse indicates whether the bit configuration eFuses are programmed.	NA	0	If BOOT_MODE[1:0] = 0b10: <ul style="list-style-type: none">• 0—The bits of the SBMR are overridden by the GPIO pins.• 1—The specific bits of the SBMR are controlled by the eFUSE settings. If BOOT_MODE[1:0] = 0b00 <ul style="list-style-type: none">• 0—The BOOT configuration eFuses are not programmed yet. The boot flow jumps to the serial downloader.• 1—The BOOT configuration eFuses are programmed. The regular boot flow is performed.
SEC_CONFIG[1:0]	SEC_C ONFIG[0] - NXP	Security Configuration, as defined in Boot security settings	NA	01	00—Reserved 01—Open (allows any program image, even if the authentication fails)

Table continues on the next page...

**Table 3-3. Boot eFUSE descriptions
(continued)**

Fuse	Config uratio n	Definition	GPIO ¹	Shipped value	Settings ²
	SEC_C ONFIG[1] - OEM				1x—Closed (The program image executes only if authenticated)
FIELD_RETURN	OEM	Enables the NXP reserved modes.			0—The NXP reserved modes are enabled/disabled based on the DIR_BT_DIS value. 1—The NXP reserved modes are enabled.
SRK_HASH[255:0]	OEM	256-bit hash value of the super root key (SRK_HASH)	NA	0	Settings vary—used by HAB
UNIQUE_ID[63:0]	NXP	Device Unique ID, 64-bit UID	NA	Unique ID	Settings vary—used by HAB
BOOT_CFG1[7:0]	OEM	Boot configuration 1	Yes	0	Specific to the selected boot mode
BOOT_CFG2[2:0]	OEM	Boot configuration 2	Yes	0	Specific to the selected boot mode
BOOT_CFG2[3]	OEM	Infinite Loop Enable at the start of the boot ROM. Used for debugging purposes. Ignored if the DIR_BT_DIS is 1 and FIELD_RETURN is 0.	Yes	0	0—Disabled 1—Enabled
LPB_BOOT	OEM	Low-Power Boot	No	0	
WDOG_ENABLE	OEM	Watchdog reset counter enable	No	0	0—The watchdog reset counter is disabled during the serial downloader. 1—The watchdog reset counter is enabled during the serial downloader.
PAD_SETTINGS	OEM	Override values for the SD/MMC and NAND boot modes	No	0	Override these IO PAD settings: <ul style="list-style-type: none">• PAD_SETTINGS[0]—Slew Rate• PAD_SETTINGS[3:1]—Drive Strength• PAD_SETTINGS[5:4]—Speed Settings .

1. This setting can be overridden by the GPIO settings when the BT_FUSE_SEL fuse is intact. See [GPIO Boot Overrides](#) for the corresponding GPIO pin.
2. 0 = intact fuse and 1= blown fuse

3.4.2 GPIO boot overrides

This table provides a list of the GPIO boot overrides:

Table 3-4. GPIO Boot Overrides

Package Pin	Direction on reset	eFuse
GPIO_AD_B0_04/BOOT_MODE0	Input	Boot Mode selection
GPIO_AD_B0_05/BOOT_MODE1	Input	
GPIO_B0_04	Input	BOOT_CFG1[0]
GPIO_B0_05	Input	BOOT_CFG1[1]
GPIO_B0_06	Input	BOOT_CFG1[2]
GPIO_B0_07	Input	BOOT_CFG1[3]
GPIO_B0_08	Input	BOOT_CFG1[4]
GPIO_B0_09	Input	BOOT_CFG1[5]
GPIO_B0_10	Input	BOOT_CFG1[6]
GPIO_B0_11	Input	BOOT_CFG1[7]
GPIO_B0_12	Input	BOOT_CFG2[0]
GPIO_B0_13	Input	BOOT_CFG2[1]
GPIO_B0_14	Input	BOOT_CFG2[2]
GPIO_B0_15	Input	BOOT_CFG2[3]

NOTE

Refer to the Fusemap chapter for more information on fuses mapped to the GPIO pins.

The input pins provided are sampled at boot, and can be used to override the corresponding eFUSE values, depending on the setting of the BT_FUSE_SEL fuse.

3.4.3 Device Configuration Data (DCD)

The DCD is the configuration information contained in the program image (external to the ROM) that the ROM interprets to configure various on-chip peripherals. See [Device Configuration Data \(DCD\)](#) for more details on DCD.

3.5 Device initialization

This section describes the details of the ROM and provides the initialization details.

This includes details on:

- The ROM memory map
- The RAM memory map
- On-chip blocks that the ROM must use or change the POR register default values
- Clock initialization

- Enabling the L1 I/D cache
- Exception handling and interrupt handling

3.5.1 Internal ROM/RAM memory map

These figures show the internal ROM and RAM memory map:

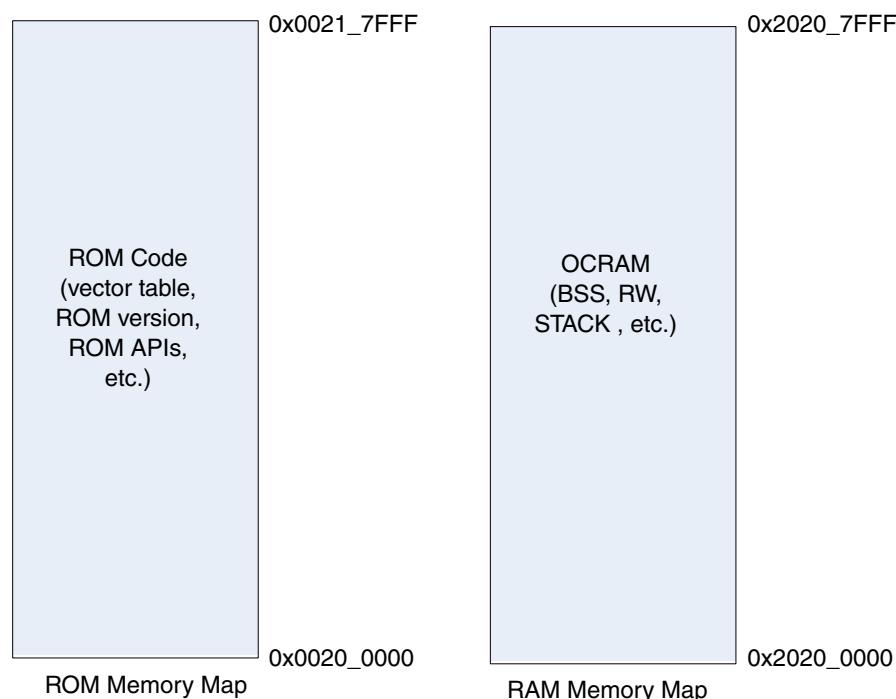


Figure 3-2. Internal ROM and RAM memory map

NOTE

The entire OCRAM region can be used freely after the boot.

3.5.2 Boot block activation

The boot ROM affects a number of different hardware blocks which are activated and play a vital role in the boot flow.

The ROM configures and uses the following blocks (listed in an alphabetical order) during the boot process. Note that the blocks actually used depend on the boot mode and the boot device selection:

Device initialization

Block	Description
CCM	Clock Control Module
FlexSPI	Flexible SPI Interface which supports serial NOR, Serial NAND and serial RAM devices
OCOTP	On Chip One Time Programmable Controller containing the eFUSES
IOMUXC	I/O Multiplexer Control which allows the GPIO use to override the eFUSE boot settings
IOMUX GPR	I/O Multiplexer control General Purpose Registers
LPSPI	Low Power SPI interface which supports serial NOR/EEPROM devices
SEMC	Smart External Memory Controller which supports parallel NOR and RAWNAND (SLC) devices
SNVS	Secure Non- Volatile Storage
SRC	System Reset Controller
USB	Used for Serial download of a boot device provisioning program
uSDHC	Ultra-Secure Digital Host Controller
WDOG 1	WatchDog Timer

3.5.3 Clocks at boot time

The table below shows the various clocks and their sources used by the ROM.

Table 3-5. Normal Frequency Clock Configurations

BOOT_FREQ(0x460[2])	LPB_BOOT(0x470[22:21])	Core Clock Frequency(MHz)
0	0	396
	1	198
	2	99
	3	49.5
1	0	528
	1	264
	2	132
	3	66

Following reset, the ARM core has access to all peripherals. The ROM code will disable the clocks listed in the following table:

Table 3-6. List of Disabled Clocks

Clock Name
CCM_CCGR0_CG6
CCM_CCGR0_CG7
CCM_CCGR0_CG8
CCM_CCGR0_CG9

Table continues on the next page...

Table 3-6. List of Disabled Clocks (continued)

Clock Name
CCM_CCGR0(CG10)
CCM_CCGR0(CG12)
CCM_CCGR0(CG13)
CCM_CCGR0(CG14)
CCM_CCGR1(CG0)
CCM_CCGR1(CG1)
CCM_CCGR1(CG2)
CCM_CCGR1(CG3)
CCM_CCGR1(CG4)
CCM_CCGR1(CG5)
CCM_CCGR1(CG6)
CCM_CCGR1(CG8)
CCM_CCGR1(CG12)
CCM_CCGR2(CG1)
CCM_CCGR2(CG3)
CCM_CCGR2(CG4)
CCM_CCGR2(CG5)
CCM_CCGR2(CG14)
CCM_CCGR2(CG15)
CCM_CCGR3(CG0)
CCM_CCGR3(CG1)
CCM_CCGR3(CG3)
CCM_CCGR3(CG5)
CCM_CCGR3(CG10)
CCM_CCGR3(CG11)
CCM_CCGR3(CG12)
CCM_CCGR3(CG13)
CCM_CCGR4(CG8)
CCM_CCGR4(CG9)
CCM_CCGR4(CG10)
CCM_CCGR4(CG11)
CCM_CCGR4(CG12)
CCM_CCGR4(CG13)
CCM_CCGR4(CG14)
CCM_CCGR4(CG15)
CCM_CCGR5(CG1)
CCM_CCGR5(CG3)
CCM_CCGR5(CG4)
CCM_CCGR5(CG7)

Table continues on the next page...

Table 3-6. List of Disabled Clocks (continued)

Clock Name
CCM_CCGR5(CG9)
CCM_CCGR5(CG10)
CCM_CCGR5(CG11)
CCM_CCGR5(CG12)
CCM_CCGR5(CG13)
CCM_CCGR6(CG0)
CCM_CCGR6(CG1)
CCM_CCGR6(CG2)
CCM_CCGR6(CG5)
CCM_CCGR6(CG6)
CCM_CCGR6(CG7)
CCM_CCGR6(CG8)
CCM_CCGR6(CG12)
CCM_CCGR6(CG13)
CCM_CCGR6(CG14)
CCM_CCGR6(CG15)

3.5.4 Enabling Caches

The boot ROM includes a feature that enables the caches to improve the boot speed.

L1 instruction cache is enabled at the start of image download. And, the L1 data cache is enabled at the start of image authentication.

By default, the L1-ICache and DCache are enabled by ROM. However, there are fuse bits that can be programmed for ROM not to enable the L1 I/DCache at boot.

The Cache features are controlled by the BT_ICACHE_DSIABLE fuse and BT_DCACHE_DISABLE fuses. By default, both fuses are not blown meaning the ROM uses the L1-ICache and L1-DCache of the ARM core. This improves the performance of the HAB signature verification software.

3.5.5 Exception handling

The exception vectors (interrupt vectors/vector table) are located at the start of ROM. During the boot phase, CPU loads the SP and PC from exception vectors and then boot to ROM.

After boot the program image can relocate the vectors as required.

3.5.6 Interrupt handling during boot

No special interrupt-handling routines are required during the boot process. The interrupts are disabled during the boot ROM execution and may be enabled in a later boot stage.

3.5.7 Persistent bits

Some modes of the boot ROM require the registers that keep their values after a warm reset. The SRC General-Purpose registers are used for this purpose.

See this table for persistent bits list and description:

Table 3-7. Persistent bits

Bit name	Bit location	Description
PERSIST_SECONDARY_BOOT	SRC_GPR10[30]	This bit identifies which image must be used—primary and secondary. Used only for eMMC/SD boot.
PERSISTENT_ENTRY0[31:0]	SRC_GPR1[31:0]	Holds the entry function for the CPU0 to wake up from the low-power mode.
PERSISTENT_ARG0[31:0]	SRC_GPR2[31:0]	Holds the argument of entry function for the CPU0 to wake up from the low-power mode.
PERSIST_REDUNDANT_BOOT	SRC_GPR10[27:26]	This field identifies which image must be used - 0/1/2/3. Used for both SPI NAND and SLC raw NAND devices.

3.6 Boot devices (internal boot)

The chip supports these boot flash devices:

- Serial NOR flash via FlexSPI Interface
- Serial NAND Flash via FlexSPI Interface
- Parallel NOR flash with the Smart External Memory Controller (SEMC), located on CS0, 16-bit bus width.
- NAND Flash with SEMC interface, located on CS0, 8-bit/16-bit bus width.
- SD/MMC/eSD/SDXC/eMMC4.4 via uSDHC interface, supporting high capacity cards
- Serial NOR/EEPROM boot via LPSPI

Boot devices (internal boot)

The selection of the external boot device type is controlled by the BOOT_CFG1[7:4] eFUSES. See this table for more details:

Table 3-8. Boot device selection

BOOT_CFG1[7:4]	Boot device
0000b	Serial NOR boot via FlexSPI
01xxb	SD Boot via uSDHC
10xxb	eMMC/MMC boot via uSDHC
001xb	SLC NAND boot via SEMC
0001b	Parallel NOR boot via SEMC
11xxb	Serial NAND boot via FlexSPI

3.6.1 Serial NOR Flash Boot via FlexSPI

3.6.1.1 Serial NOR eFUSE Configuration

Table 3-9. Fuse definition for Serial NOR over FlexSPI

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG1[1]	OEM	Encrypted XIP	Yes	0	0 – Disabled 1 - Enabled
BOOT_CFG1[3:2]	OEM	Hold time before read from device	Yes	0	0 – 500us 1 – 1ms 2 – 3ms 3 – 10ms
BOOT_CFG1[7:4]	OEM	Boot device selection	Yes	0	0 – Serial NOR device is selected as boot device
BOOT_CFG2[2:0]	OEM	Flash Type	Yes	0	000b-Device supports 3B read by default 001b-Device supports 4B read by default 010b-HyperFlash 1V8 011b-HyperFlash 3V3 100b-MXIC Octal DDR 101b-Micron Octal DDR

Table continues on the next page...

Table 3-9. Fuse definition for Serial NOR over FlexSPI (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					111b-QSPI device supports 3B read by default (on secondary pinmux option)
BOOT_CFG2[3]	BOOT_CFG2[3]	Infinite-Loop (Debug use only)	Yes	0	0 - Disable 1 - Enable

3.6.1.2 FlexSPI Serial NOR Flash Boot Operation

The Boot ROM attempts to boot from Serial NOR flash if the BOOT_CFG1 [7:4] fuses are programmed to 0b'0000 as shown in the Serial NOR eFUSE Configuration table, then the ROM will initialize FlexSPI1 interface. FlexSPI interface initialization is a two-step process.

The ROM expects the 512-byte FlexSPI NOR configuration parameters as explained in next section to be present at offset 0 in Serial NOR flash. The ROM reads these configuration parameters using the read command specified by BOOT_CFG2[2:0] with Serial clock operating at 30 MHz.

In the second step, ROM configures FlexSPI1 interface with the parameters provided in configuration block read from Serial NOR flash and starts the boot procedure. Refer to [Table 3-23](#) for details regarding FlexSPI configuration parameters and to the FlexSPI NOR boot flow chart for detailed boot flow chart of FlexSPI NOR.

Both booting an XIP and non XIP image are supported from Serial NOR Flash. For XIP boot, the image has to be built for FlexSPI address space and for non XIP the image can be built to execute from Internal RAM.

3.6.1.3 FlexSPI NOR boot flow chart

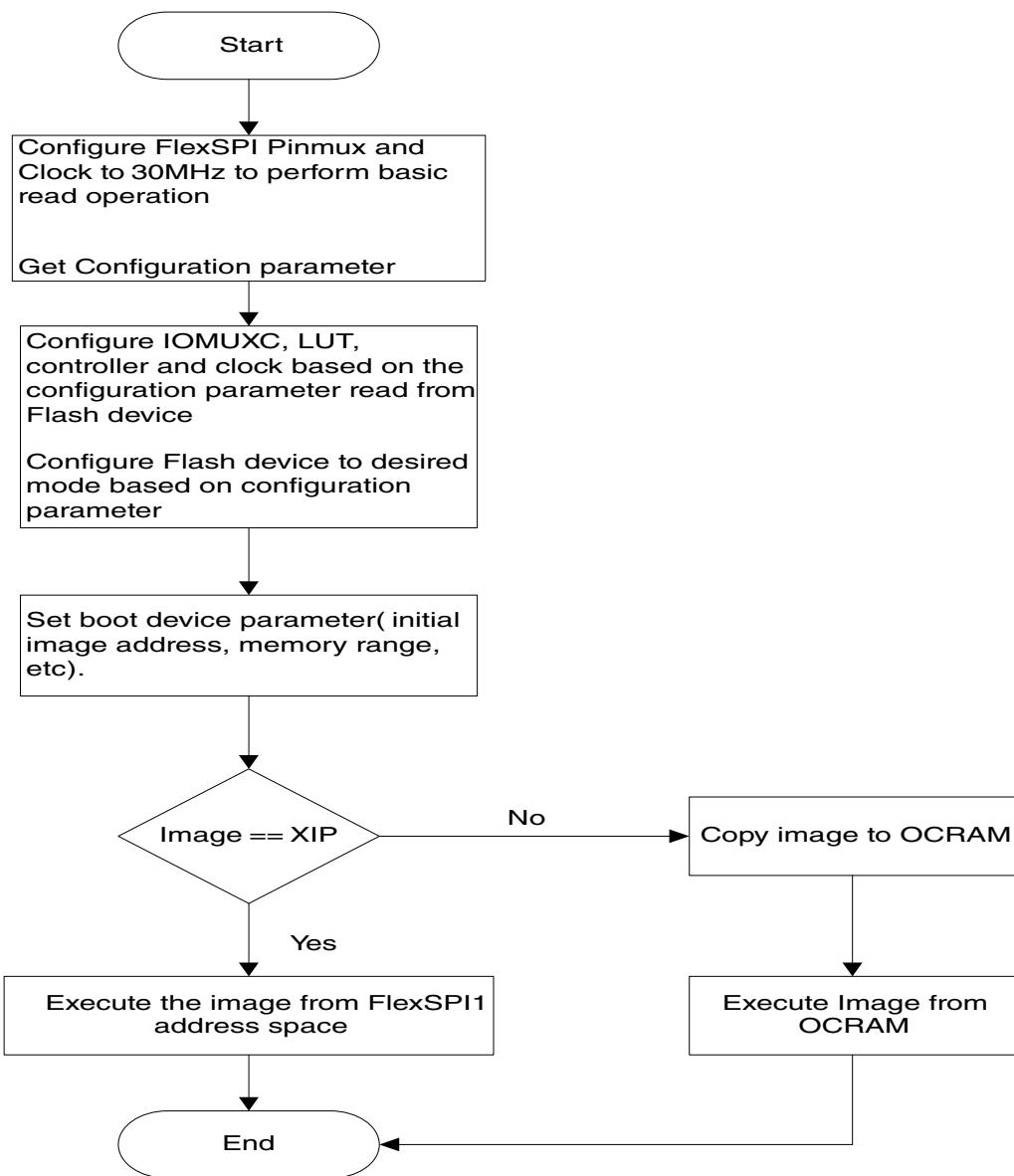


Figure 3-3. FlexSPI NOR boot flow

3.6.2 Encrypted XIP and Flash Access Protection

3.6.2.1 Encrypted XIP on Serial NOR via FlexSPI Interface

The ROM supports eXecute In Place (XIP) on the Serial NOR flash device directly with On-the-fly decryption feature (using AES-CTR-128 or AES-ECB-128) powered by BEE controller.

The BootROM supports two separate encrypted regions using two separate AES Keys. Before doing Encrypted XIP, the BootROM needs to set the BEE controller correctly, the configurable parameters are organized as Protection Region Descriptor Block (PRDB), the entire PRDB is encrypted using AES-CBC-128 mode with the AES KEY and IV in a Key Info Block (KIB). The KIB is encrypted as Encrypted KIB (EKIB) using the AES key provisioned in eFUSE (SW_GP2) or derived from OTPMK. The BootROM decrypts KIB using AES ECB-128 mode, up to 2 EKIBs are supported, EKIB0 is located at offset 0x400 and KIB1 is located at offset 0x800. After getting KIB, the BootROM uses the AES key and IV in the KIB to decrypt PRDB from Encrypted PRDB(EPRDB), up 2 EPRDBs are supported, EPRDB0 is located at offset 0x480 and EPRDB1 is located at 0x880, both are encrypted using AES-CBC-128 mode.

Table 3-10. eFUSE Configuration for Encrypted XIP

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG [1] 0x450[1]	OEM	Encrypted XIP Enable	Yes	0	0 - Encrypted XIP is bypassed 1 - Encrypted XIP is enabled
BEE_KEY0_SEL 0x460[13:12]	OEM	AES Key selection for BEE_KEY0	No	0	0 - From register 1 - Reserved 2 - From OTPMK [255:128] 3 - FROM SW-GP2
BEE_KEY1_SEL 0x460[15:14]	OEM	AES Key Selection for BEE_KEY1	No	0	0 - From register 1 - Reserved 2 - From OTPMK [255:128] 3 - FROM SW-GP2

NOTE

- All above FUSES are valid only if the BEE controller is present in this SoC.
- AES Key selection from register is NOT supported by ROM.

3.6.2.2 FlexSPI NOR Image layout

The FlexSPI NOR image layout is as shown in the table [Table 3-11](#). Both EPRDB0 and EPRDB1 are optional, and they are required only if corresponding eFUSE bits are enabled.

PRDB1 takes precedence over PRDB0, this design is for the use case that the Independent Design House(IDH) uses this block to protect their intellectual property before shipping products to end customers.

Table 3-11. Boot Image Layout

Offset	Description
0x0000 - 0x01FF	FlexSPI Serial NOR Configuration Block
0x0200 - 0x03FF	Reserved
0x0400 - 0x041F	Encrypted Key Info Block 0 (EKIB0)
0x0420 - 0x047F	Reserved
0x0480 - 0x057F	Encrypted Protection Region Descriptor Block 0(EPRDB0)
0x0580 - 0x07FF	Reserved
0x0800 - 0x081F	Encrypted Key Info Block 1(EKIB1) Only the first 32bytes are used, the remaining bytes are reserved for future use.
0x0820 - 0x087F	Reserved
0x0880 - 0x097F	Encrypted Protection Region Descriptor Block 1 (EPRDB1)
0x0980 - 0x0FFF	Reserved
0x1000 - Flash end	Boot image (IVT, DCD, Application, Execute-only library, etc.)

3.6.2.3 Key Info Block

Key Info Block consist of 128-bit AES Key and 128-bit Initial Vector.

Table 3-12. Key Info Block

Offset	Field	Description
0x00 - 0x0F	AES-128 KEY	128-bit AES KEY used for EPRDB encryption and decryption
0x10 - 0x1F	Initial Vector	128-bit Initial Vector used for EPRDB encryption and decryption

NOTE

- KIB1 is encrypted to EKIB1 using BEE_KEY1 provisioned in eFUSE
- KIB0 is encrypted to EKIB0 using BEE_KEY0 provisioned in eFUSE

3.6.2.4 Protection Region Descriptor Block (PRDB)

PRDB is a 128-byte crypto block using AES-ECB mode and the AES Key provisioned in eFUSE. The plaintext of PRDB is shown in the table below.

Table 3-13. PRDB

Offset	Field	Description
0x000 - 0x003	tagl	Must equal to 0x5F474154 (ASCII: "TAG_")
0x004 - 0x007	tagh	Must equal to 0x52444845 (ASCII: "EHDR")
0x008 - 0x00b	version	[31:24] 'V' [23:16] Major version [15:08] Minor version [07:00] Bug fix Must equal to 0x56010000 for i.MX RT Family
0x00C - 0x00F	fac_region_count	Flash Access Controlled (FAC) Region within this encrypted region. Valid value: 1-3
0x010-0x04F	encrypt_region_info	Information for encrypted region, see Table 3-14
0x050-0x0CF	fac_regions	PRDB supports up to 4 Flash access controlled regions. The first 3 regions are available for application use, the last one is Reserved. For FAC regions definitions, see Table 3-15
0x0D0-0x0FF	Reserved	Reserved for future use

Table 3-14. Information for encrypted region

Offset	Field	Description
0x00 - 0x03	start	Absolute start address of encrypted region, must be 4KB aligned
0x04 - 0x07	end	Absolute end address of encrypted region, must be 4KB aligned
0x08 - 0x0b	mode	AES encryption mode 0 - AES-ECB (128) 1 - AES-CTR (128), it is the recommend mode for Encrypted XIP
0x0C - 0x0F	lock_option	Region Lock options 0 - All regions are unlocked 1 - All FAC regions that belong to BEE Region1 are locked 2 - All RAC regions that belong to BEE Region0 are locked

Table continues on the next page...

Table 3-14. Information for encrypted region (continued)

		3 - Both BEE regions are locked, All FAC regions defined in PRDB are locked.
0x10 - 0x1F	counter	AES-CTR Counter, Valid only if mode is 1 Only the upper 96bits are configurable, the lower 32bit must be 0. Note: During encryption and decryption, the nonce/is as show in Table 3-15 ("FAC Regions" table).
0x20-0x3F	Reserved	Reserved for future use.

Table 3-15. FAC Regions

Offset	Field	Description
0x00-0x03	start	Absolute start region of FAC protection region, must be 1KB aligned
0x04-0x07	end	Absolute end region of FAC protected region, must be 1KB aligned
0x08-0x0B	mode	Protected mode. Supported options: 0/1. 0 - Flash Access is not restricted 1 - M7 Debug is disabled Others – Unpredictable behavior
0x0C-0x1F	Reserved	Reserved for future use

127:32	31:28	27:0
Counter [127:32]	0	FAC Region Start [31:4]

NOTE

- PRDB0 is encrypted using AES_KEY and IV in EKIB0, while the encrypted region defined in PRDB0 is encrypted by the key source specified in BEE_KEY0_SEL.
- PRDB1 is encrypted using AES_KEY and IV in EKIB1, while the encrypted region defined in PRDB1 is encrypted by the key source specified in BEE_KEY1_SEL.

3.6.2.5 Encrypted XIP Flow

The encrypted XIP flow is shown in the diagram below:

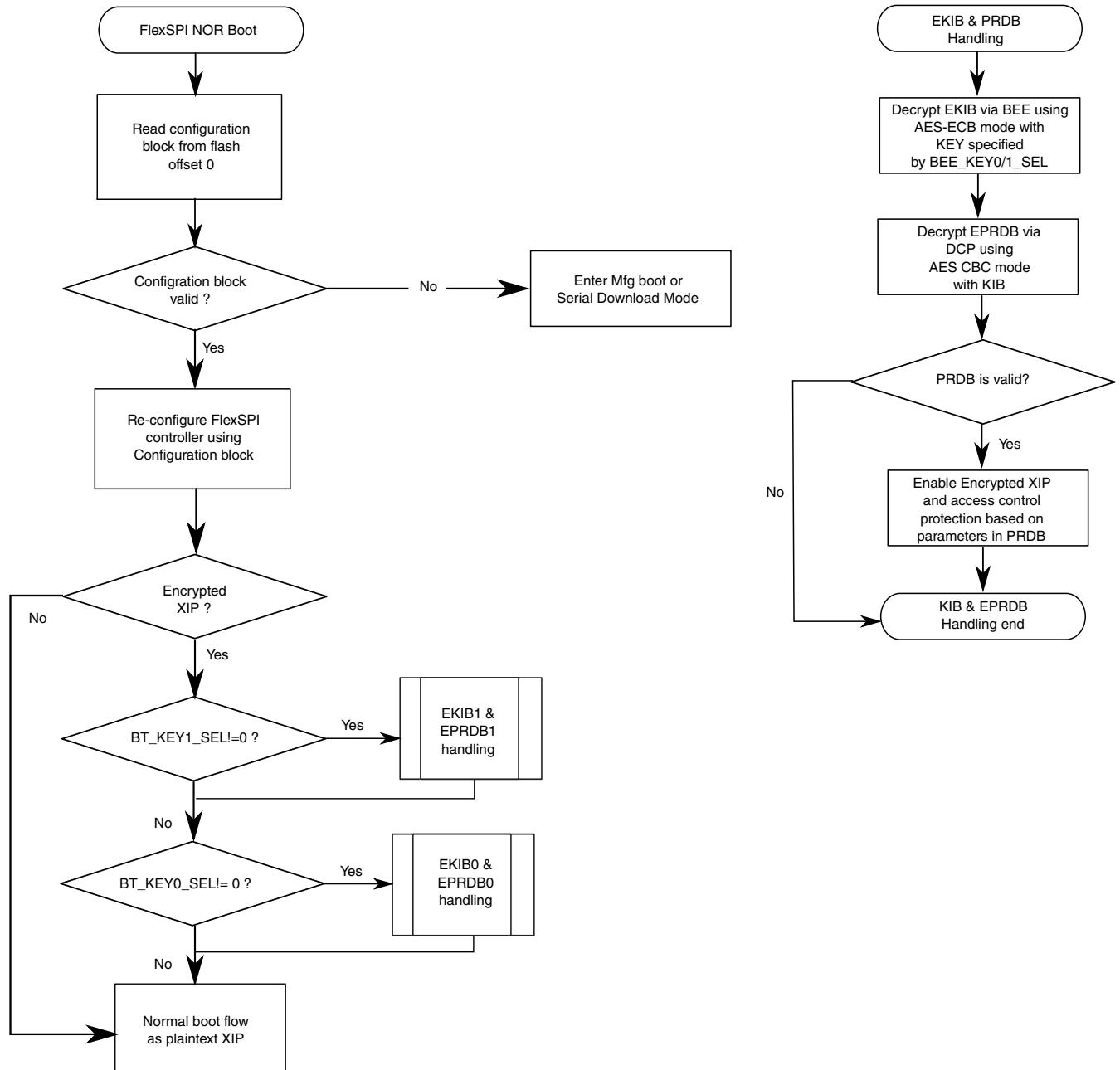


Figure 3-4. Encrypted XIP Flow

3.6.2.6 Data and Key Endianness

SoCs belongs to i.MX RT family only support little-endian mode, to program correct AES key and generate correct encrypted boot image, Endian conversion must be taken into consideration.

Boot devices (internal boot)

- The AES key for BEE must be programmed using big-endian mode, for example, if the AES KEY is {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff}, it should be programmed in eFUSE following this order: 0xccddeeef, 0x8899aabb, 0x44556677, 0x00112233.
- The data to be encrypted are little-endian ordered by default, so, no swap operations are needed.
- The nonce/counter must be swapped before performing encryption using AES-CTR-128 mode. For example, if the counter in PRDB is {0x06000100, 0x44332211, 0x88776655, 0xccbb99aa}, the actual CTR for encryption is {0xcc, 0xbb, 0xaa, 0x99, 0x88, 0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11, 0x06, 0x00, 0x01, 0x00}.

3.6.3 Serial NAND Flash Boot over FlexSPI

The boot ROM supports a number of Serial NAND Flash devices from different vendors. The Embedded Error Correction and Control (ECC) module in SPI NAND devices are used to detect and correct the errors.

3.6.3.1 Serial NAND eFUSE Configuration

The boot ROM determines the configuration of external Serial NAND flash by parameters, either provided by eFUSE, or sampled on GPIO pins during boot. See below table for parameters details.

Table 3-16. Fuse definition for Serial NAND over FlexSPI

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG1[1:0]	OEM	Search Stride for FCB and DBBT Search strides in terms of page	Yes	0	0 - 64 1 - 128 2 - 256 3 - 32
BOOT_CFG1[3:2]	OEM	Hold Time before access to Serial NAND	Yes	0	0 - Hold time determined by Read Status command 1 - 500us 2 - 1ms 3 - 3ms
BOOT_CFG1[4]	OEM	Column address width	Yes	0	0 - 12 bits 1 - 13 bits

Table continues on the next page...

Table 3-16. Fuse definition for Serial NAND over FlexSPI (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG1[5]	OEM	Default safe communication frequency	Yes	0	0 – High Speed (50MHz) 1 – Low Speed (30MHz)
BOOT_CFG1[7:6]	OEM	Primary boot device selection	Yes	0	00 – Serial NOR 11 – Serial NAND
BOOT_CFG2[0]	OEM	Boot Search Count of FCB and DBBT	Yes	0	0 - 1 1 - 2
BOOT_CFG2[2:1]	OEM	CS de-asserted interval between two commands	Yes	0	0 – 100ns 1 – 200ns 2 – 400ns 3 – 50ns
BOOT_CFG2[3]	OEM	Infinite-Loop: (Debug USE only)	Yes	0	0 - Disable 1- Enable
0x6E0[7]	OEM	SPI NAND BOOT - Override Busy Offset	No	0	0 – Use default busy bit offset 0 1 – Override default busy bit offset using Busy bit offset
0x6E0[3:0]	OEM	SPI NAND BOOT - Busy Bit offset	No	0	
0x6E0[13:8]	OEM	SPI NAND Boot - Page Read Time (in terms of micro seconds)	No	0	
0x6E0[23:16]	OEM	Page Read Command	No	0	Available only if it is not 0
0x6F0[31:24]	OEM	Cache Read command	No	0	Available only if it is not 0

NOTE

BOOT_CFGx sampled on GPIO pins depends on BT_FUSE_SEL setting.

3.6.3.2 FlexSPI NAND Flash Boot Flow and Boot Control Blocks (BCB)

There are two BCB data structures:

- FCB
- DBBT

Boot devices (internal boot)

As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a Firmware Configuration Block (FCB) that contains the optimum NAND timings, page address of Discovered Bad Block Table (DBBT) Search Area and start page address of primary and secondary firmware.

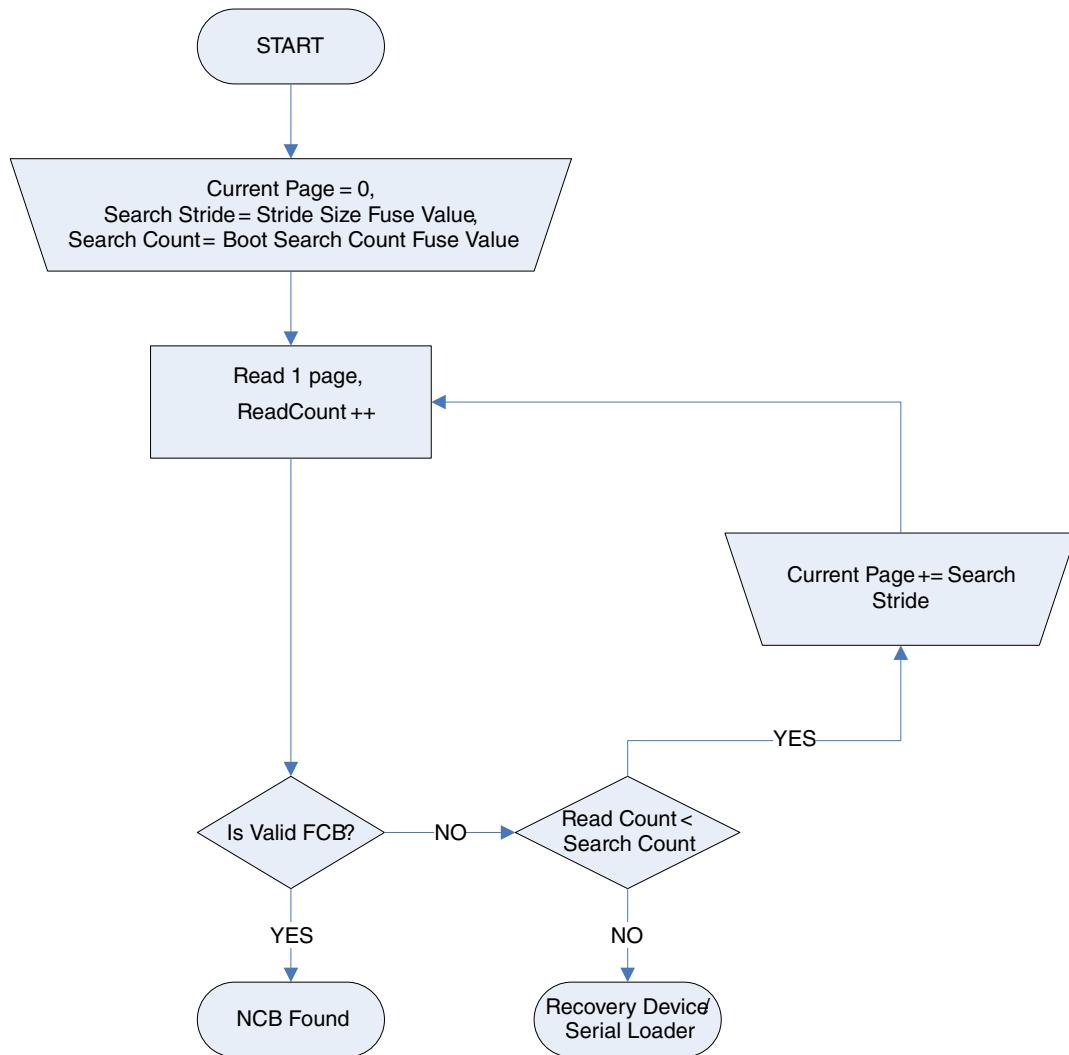
The built-in HW ECC in Serial NAND device is used during data read, the FCB data structure is protected using CRC checksum. Driver reads raw 2048 bytes of first sector and runs through CRC check that determines whether FCB data is valid or not.

If the FCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails, or the fingerprints do not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next BCB until SearchCount pages have been read.

If search fails to find a valid FCB, the NAND driver responds with an error and the boot ROM enters into serial download mode.

The FCB contains the page address of DBBT Search Area, and the page address for primary and secondary boot images. DDBT is searched in DDBT Search Area just like how FCB is searched. After the FCB is read, the DDBT is loaded, and the primary or secondary boot image is loaded using starting page address from FCB.

[Figure 3-5](#) shows the state diagram of FCB search.

**Figure 3-5. FCB Search Flow**

Once FCB is found, the boot ROM searches for the Discovered Bad Blocks Table (DBBT). If DBBT Search Area is 0 in FCB, then ROM assumes that there are no bad blocks on NAND device boot area. See [Figure 3-6](#) for the DBBT search flow.

Boot devices (internal boot)

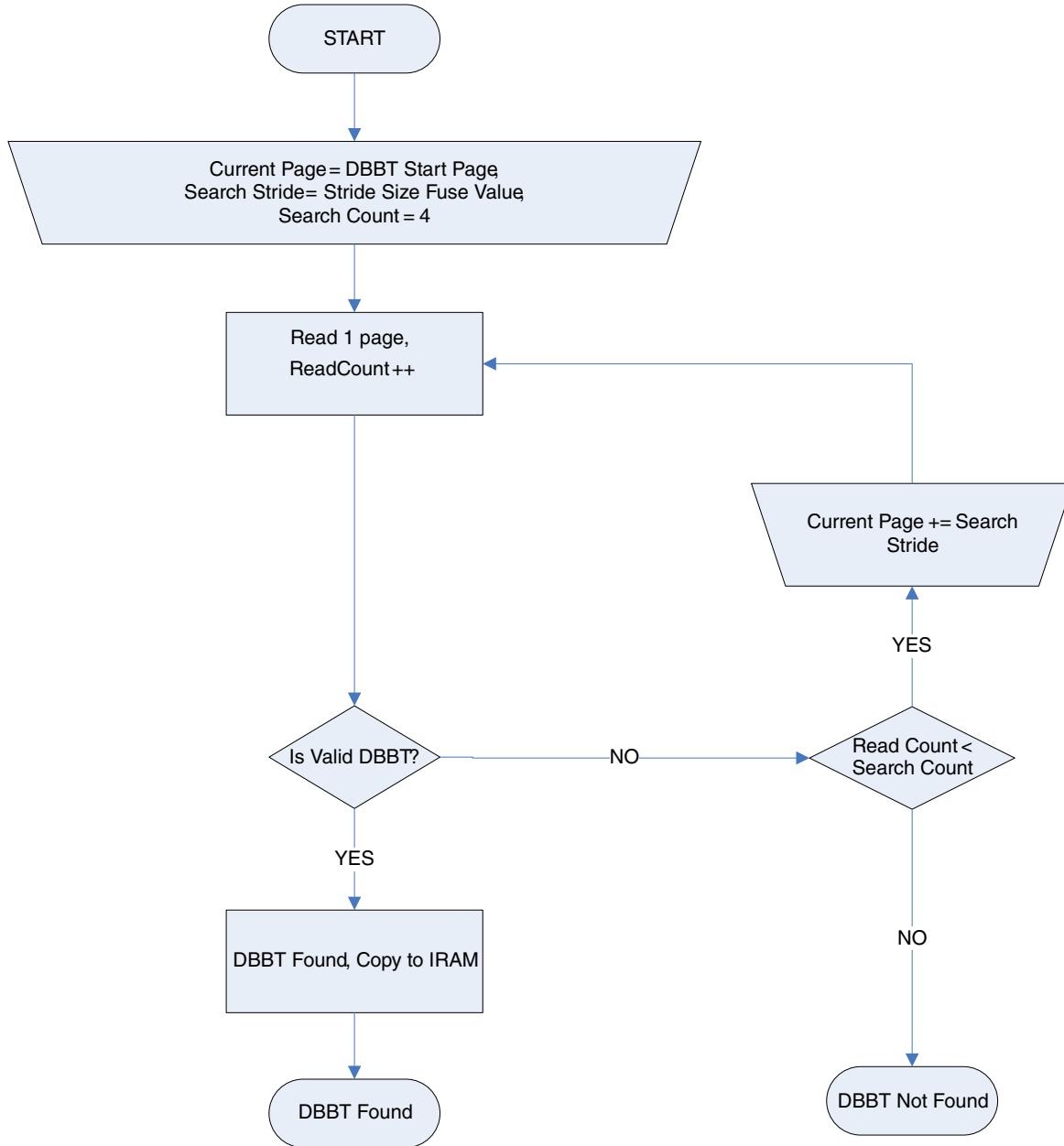


Figure 3-6. DBBT Search Flow

The BCB search and load function also monitors the ECC correction threshold and sets the PERSIST_BLOCK_REWRITE persistent bit if the threshold exceeds the maximum ECC correction ability.

If during primary image read there is a page with a number of errors higher than ECC can correct, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit and perform SW reset (After SW reset, secondary image is used).

If during secondary image read there is a page with number of errors higher than ECC can correct, the boot ROM goes to serial loader.

3.6.3.3 Firmware Configuration Block

The FCB is at the first page in the first good block. The FCB should be present at each search stride of the search area.

The search area contains copies of the FCB at each stride distance, in case the first Serial NAND block becomes corrupted, the ROM will find its copy in the next Serial NAND block. The search area should span over at least two Serial NAND blocks. The location information for DBBT search area and images are all specified in the FCB. Table 25-9 show the Flash Control Block Structure.

Table 3-17. Flash Control Block Structure

Name	Offset	Size Bytes	Description									
crcChecksum	0x000	4	Cheksum									
fingerprint	0x004	4	0x4E46_4342 ASCII: "NFCB"									
version	0x008	4	0x0000_0001									
DBBTSearchStartPage	0x00C	4	Start Page address for bad block table search area									
searchStride	0x010	2	Search stride for DBBT and FCB search. Not used by ROM Max value is 8.									
searchCount	0x012	2	Copies of DBBT and FCB. Not used by ROM, max value is 8.									
firmwareCopies	0x014	4	Firmware copies Valid range 1-8.									
Reserved	0x018	40	Reserved for future use Must be set to 0.									
firmwareInfoTable	0x40	64	This table consists of (up to 8 entries): <table border="1" data-bbox="1142 1534 1468 1845"> <thead> <tr> <th>Field</th><th>Size</th><th>Description</th></tr> </thead> <tbody> <tr> <td>StartPage</td><td>4</td><td>Start page of this firmware</td></tr> <tr> <td>pageCount</td><td>4</td><td>Pages in this firmware</td></tr> </tbody> </table>	Field	Size	Description	StartPage	4	Start page of this firmware	pageCount	4	Pages in this firmware
Field	Size	Description										
StartPage	4	Start page of this firmware										
pageCount	4	Pages in this firmware										

Table continues on the next page...

**Table 3-17. Flash Control Block Structure
(continued)**

Name	Offset	Size Bytes	Description
			NOTE: The StartPage must be the first page of a NAND block.
Reserved	0x080	128	Reserved Must be set to 0
spiNandConfigBlock	0x100	512	Serial NAND configuration block over FlexSPI
Reserved	0x300	256	Must be set to 0

NOTE

1. The “crcChecksum” is calculated with an MPEG2 variant of CRC-32. See [Table 3-18](#) for more details.
2. The “crcChecksum” calculation starts from fingerprint to the end of FCB, 1020 bytes in total.
3. The “spiNandConfigBlock” is FlexSPI NAND configuration block which consists of common FlexSPI memory configuration block and Serial NAND specified configuration parameters.

Table 3-18. CRC-32 variant algorithm

Property	Description
Width	32 bits
Polynomial	0x04C11BD7
Init Value	0xFFFFFFFF
Reflect in	False
Reflect Out	False
XOR Out	0x00000000

3.6.3.4 Discovered Bad Blocks Table (DBBT)**Table 3-19. DBBT Structure**

Name	Offset	Size in Bytes	Description
crcChecksum	0x000	4	Checksum
Fingerprint	0x004	4	32-bit word with a value of 0x4442_4254, in ascii “DBBT”

Table continues on the next page...

Table 3-19. DBBT Structure (continued)

Name	Offset	Size in Bytes	Description
Version	0x008	4	32-bit version number, this version of DBBT is 0x00000001
-	0x00C	4	Reserved
badBlockNumber	0x010	4	Number of bad blocks
Reserved	0x014	12	Must be filled with 0x00s
Bad Block entries	0x020-0x41F	256	Bad Block entries, only the first “badBlockNumber” entries are valid, the remaining entries must be filled with 0x00s

NOTE

1. Maximum bad block number is 256.
2. The "crcChecksum" is calculated with the same algorithm as the one in FCB, from Fingerprint to the end of DBBT, 1052 bytes in total.

3.6.3.5 Bad block handling in ROM

During the firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block.

If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block checked.

3.6.4 Serial NOR and NAND configuration based on FlexSPI interface

The ROM SW supports Serial NOR and Serial NAND based on FlexSPI module, using a 448-bytes common FlexSPI configuration block and several specified parameters for Serial NOR and Serial NAND respectively. See below sections for more details.

3.6.4.1 FlexSPI Configuration Block

FlexSPI Configuration block consists of parameters regarding specific Flash devices including read command sequence, quad mode enablement sequence (optional), etc.

Table 3-20. FlexSPI Configuration block

Name	Offset	Size(bytes)	Description
Tag	0x000	4	0x42464346, ascii:"FCFB"
Version	0x004	4	0x56010000 / 0x56010100 [07:00] bugfix = 0 [15:08] minor = 0 / 1 [23:16] major = 1 [31:24] ascii 'V'
-	0x008	4	Reserved
readSampleClkSrc	0x00C	1	0 – internal loopback 1 – loopback from DQS pad 3 – Flash provided DQS
dataHoldTime	0x00D	1	Serial Flash CS Hold Time Recommend default value is 0x03
dataSetupTime	0x00E	1	Serial Flash CS setup time Recommended default value is 0x03
columnAdressWidth	0x00F	1	3 – For HyperFlash 12/13 – For Serial NAND, see datasheet to find correct value 0 – Other devices
deviceModeCfgEnable	0x010	1	Device Mode Configuration Enable feature 0 – Disabled 1 – Enabled
-	0x011	1	Reserved
waitTimeCfgCommands	0x013	2	Wait time for all configuration commands, unit 100us. Available for device that support v1.1.0 FlexSPI configuration block. If it is greater than 0, ROM will wait waitTimeCfgCommands * 100us for all device memory configuration commands instead of using read status to wait until these commands complete.
deviceModeSeq	0x014	4	Sequence parameter for device mode configuration Bit[7:0] - number of LUT sequences for Device mode configuration command

Table continues on the next page...

Table 3-20. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
			Bit[15:8] - starting LUT index of Device mode configuration command Bit[31:16] - must be 0
deviceModeArg	0x018	4	Device Mode argument, effective only when deviceModeCfgEnable = 1
configCmdEnable	0x01C	1	Config Command Enable feature 0 – Disabled 1 – Enabled
-	0x01D	3	Reserved
configCmdSeqs	0x020	12	Sequences for Config Command, allow 3 separate configuration command sequences.
-	0x02C	4	Reserved
cfgCmdArgs	0x030	12	Arguments for each separate configuration command sequence.
-	0x03C	4	Reserved
controllerMiscOption	0x040	4	Bit0 – differential clock enable Bit1 – CK2 enable, must set to 0 in this silicon Bit2 – ParallelModeEnable, must set to 0 for this silicon Bit3 – wordAddressableEnable Bit4 – Safe Configuration Frequency enable set to 1 for the devices that support DDR Read instructions Bit5 – Pad Setting Override Enable Bit6 – DDR Mode Enable, set to 1 for device supports DDR read command
deviceType	0x044	1	1 – Serial NOR 2 – Serial NAND
sflashPadType	0x045	1	1 – Single pad 2 – Dual pads 4 – Quad pads 8 – Octal pads
serialClkFreq	0x046	1	Chip specific value, for this silicon

Table continues on the next page...

Table 3-20. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
			1 – 30 MHz 2 – 50 MHz 3 – 60 MHz 4 – 75 MHz 5 – 80 MHz 6 – 100 MHz 7 – 133 MHz 8 – 166 MHz Other value: 30 MHz
lutCustomSeqEnable	0x047	1	0 – Use pre-defined LUT sequence index and number 1 - Use LUT sequence parameters provided in this block
-	0x048	8	Reserved
sflashA1Size	0x050	4	For SPI NOR, need to fill with actual size For SPI NAND, need to fill with actual size * 2
sflashA2Size	0x054	4	The same as above
sflashB1Size	0x058	4	The same as above
sflashB2Size	0x05C	4	The same as above
csPadSettingOverride	0x060	4	Set to 0 if it is not supported
sclkPadSettingOverride	0x064	4	Set to 0 if it is not supported
dataPadSettingOverride	0x068	4	Set to 0 if it is not supported
dqsPadSettingOverride	0x06C	4	Set to 0 if it is not supported
timeoutInMs	0x070	0	Maximum wait time during read busy status 0 – Disabled timeout checking feature Other value – Timeout if the wait time exceeds this value.
commandInterval	0x074	4	Unit: ns Currently, it is used for SPI NAND only at high frequency
dataValidTime	0x078	4	Time from clock edge to data valid edge, unit ns. This field is used when the FlexSPI Root clock is less than 100 MHz and the read sample clock source is device provided DQS signal without CK2 support.

Table continues on the next page...

Table 3-20. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
			[31:16] data valid time for DLLB in terms of 0.1 ns [15:0] data valid time for DLLA in terms of 0.1 ns
busyOffset	0x07C	2	busy bit offset, valid range : 0-31
busyBitPolarity	0x07E	2	0 – busy bit is 1 if device is busy 1 – busy bit is 0 if device is busy
lookupTable	0x080	256	Lookup table
lutCustomSeq	0x180	48	Customized LUT sequence, see below table for details.
	0x1B0	16	Reserved for future use

Note:

1. To customize the LUT sequence for some specific device, users need to enable “lutCustomSeqEnable” and fill in corresponding “lutCustomSeq” field specified by command index below.
2. For Serial (SPI) NOR, the pre-defined LUT index is as follows:

Table 3-21. LUT sequence definition for Serial NOR

Command Index	Name	Index in lookup table	Description
0	Read	0	Read command Sequence
1	ReadStatus	1	Read Status command
2	WriteEnable	3	Write Enable command sequence
3	EraseSector	5	Erase Sector Command
4	PageProgram	9	Page Program Command
5	ChipErase	11	Full Chip Erase
6	Dummy	15	Dummy Command as needed
	Reserved	2,4,6,7,8,10,12,13,14	All reserved indexes can be freely used for other purpose

3. For Serial (SPI) NAND, the pre-defined LUT index is as follows:

Table 3-22. LUT sequence definition for Serial NAND

Command Index	Name	Index in lookup table	Description
0	ReadFromCache	0	Read from cache
1	ReadStatus	1	Read Status
2	WriteEnable	3	Write Enable
3	BlockErase	5	Erase block
4	ProgramLoad	9	Program Load
5	ReadPage	11	Read page to cache
6	ReadEccStatus	13	Read ECC Status
7	ProgramExecute	14	Program Execute
8	ReadFromCacheOdd	4	Read from Cache while page in odd plane
9	ProgramLoadOdd	10	Program Load for pages within odd blocks
	Reserved	2,6,7,8,12,15	All reserved indexes can be freely used for other purposes

NOTE

All the predefined LUT indexes are only applicable during ROM execution. User application should NOT rely on the ROM configurations except the READ command sequence.

3.6.4.2 Serial NOR configuration block (512 bytes)

Table 3-23. Serial NOR configuration block

Name	Offset	Size (Bytes)	Description
memCfg	0	448	The common memory configuration block, see FlexSPI configuration block for more details
pageSize	0x1C0	4	Page size in terms of bytes, not used by ROM
sectorSize	0x1C4	4	Sector size in terms of bytes, not used by ROM
ipCmdSerialClkFreq	0x1C8	4	Chip specific value, not used by ROM 0 – No change, keep current serial clock unchanged

Table continues on the next page...

Table 3-23. Serial NOR configuration block (continued)

Name	Offset	Size (Bytes)	Description
			1 – 30 MHz 2 – 50 MHz 3 – 60 MHz 4 – 75 MHz 5 – 80 MHz 6 – 100 MHz 7 – 133 MHz 8 – 166 MHz
Reserved	0x1CC	52	Reserved for future use

3.6.4.3 Serial NAND configuration block (512 bytes)

Table 3-24. Serial NAND configuration block

Name	Offset	Size (Bytes)	Description
memCfg	0	448	The common memory configuration block, see FlexSPI configuration block for more details
pageDataSize	0x1C0	4	Page size in terms of bytes, usually, it is 2048 or 4096
pageTotalSize	0x1C4	4	It equals to 2^{\wedge} width of column address
pagesPerBlock	0x1C8	4	Pages in one block
bypassReadStatus	0x1CC	1	0 – Read Status Register 1 – Bypass Read status register
bypassEccRead	0x1CD	1	0 – Perform ECC read 1 – Bypass ECC read
hasMultiPlanes	0x1CE	1	0 – Only 1 plane 1 – Has two planes
skippOddBlocks	0x1CF	1	0 – Read Odd blocks 1 – Skip Odd blocks
eccCheckCustomEnable	0x1D0	1	0 – Use the common ECC check command and ECC related masks 1 - Use ECC check related masks provided in this configuration block
ipCmdSerialClkFreq	0x1D1	1	Chip specific value, not used by ROM

Table continues on the next page...

Table 3-24. Serial NAND configuration block (continued)

Name	Offset	Size (Bytes)	Description
			0 – No change, keep current serial clock unchanged 1 – 30 MHz 2 – 50 MHz 3 – 60 MHz 4 – 75 MHz 5 – 80 MHz 6 – 100 MHz 7 – 133 MHz 8 – 166 MHz
readPageTimeUs	0x1D2	2	Wait time during page read, this field will take effect on if the bypassReadStatus is set to 1. NOTE: Only applicable to ROM.
eccStatusMask	0x1D4	4	ECC Status Mask
eccFailureMask	0x1D8	4	ECC Check Failure mask
blocksPerDevice	0x1DC	4	Blocks in a Serial NAND
Reserved	0x1ED	32	Reserved for future use

Below is an example of Serial NAND configuration block for Winbond W25N01GVZEIG:

```
const flexspi_nand_config_t kSerialNandCfgBlk =
{
    .memConfig =
    {
        .tag = FLEXSPI_CFG_BLK_TAG,
        .version = FLEXSPI_CFG_BLK_VERSION,
        .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackInternally,
        .dataHoldTime = 3,
        .dataSetupTime = 3,
        .columnAddressWidth = 12,
        .deviceModeCfgEnable = 1,
        .deviceModeSeq = { 1, 2 },
        .deviceType = kFlexSpiDeviceType_SerialNAND,
        .sflashPadType = kSerialFlash_4Pads,
        .serialClkFreq = kFlexSpiSerialClk_50MHz,
        .lutCustomSeqEnable = 0,
        .sflashA1Size = 128 * 1024 * 1024U * 2, // Flash size = 2 * actual data size
(exclude spare space)
        .lookupTable =
        {
            // Read cache 4 I/O
            [4 * NOR_CMD_LUT_SEQ_IDX_READ] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB,
CADDR_SDR, FLEXSPI_4PAD, 0x10),
            [4 * NOR_CMD_LUT_SEQ_IDX_READ + 1] = FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD,
0x04, READ_SDR, FLEXSPI_4PAD, 0x80),
            // Clear Status1 flag
            [4 * 2] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x1F, CMD_SDR, FLEXSPI_1PAD,
0xA0),
    
```

```

[4 * 2 + 1] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x00, STOP, FLEXSPI_1PAD,
0x00),
    // Read Page
    [4 * NAND_CMD_LUT_SEQ_IDX_READPAGE] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x13, RADDR_SDR, FLEXSPI_1PAD, 0x18),
    // Read Status
    [4 * NAND_CMD_LUT_SEQ_IDX_READSTATUS] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x0F, CMD_SDR, FLEXSPI_1PAD, 0xC0),
    [4 * NAND_CMD_LUT_SEQ_IDX_READSTATUS + 1] = FLEXSPI_LUT_SEQ(READ_SDR,
FLEXSPI_1PAD, 0x01, STOP, FLEXSPI_1PAD, 0),

    // Write Enable
    [4 * NAND_CMD_LUT_SEQ_IDX_WRITEENABLE] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x06, STOP, FLEXSPI_1PAD, 0),

    // Page Program Load 4x
    [4 * NAND_CMD_LUT_SEQ_IDX_PROGRAMLOAD] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x32, CADDR_SDR, FLEXSPI_1PAD, 0x10),
    [4 * NAND_CMD_LUT_SEQ_IDX_PROGRAMLOAD + 1] = FLEXSPI_LUT_SEQ(WRITE_SDR,
FLEXSPI_4PAD, 0x40, STOP, FLEXSPI_1PAD, 0),
    // Page Program Execute
    [4 * NAND_CMD_LUT_SEQ_IDX_PROGRAMEXECUTE] = FLEXSPI_LUT_SEQ(CMD_SDR,
FLEXSPI_1PAD, 0x10, RADDR_SDR, FLEXSPI_1PAD, 0x18),
    // Erase Sector
    [4 * NAND_CMD_LUT_SEQ_IDX_ERASEBLOCK] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0xD8, RADDR_SDR, FLEXSPI_1PAD, 0x18),
    // Read ECC status
    [4 * NAND_CMD_LUT_SEQ_IDX_READECCSTAT] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x0F, CMD_SDR, FLEXSPI_1PAD, 0xC0),
    [4 * NAND_CMD_LUT_SEQ_IDX_READECCSTAT + 1] = FLEXSPI_LUT_SEQ(READ_SDR,
FLEXSPI_1PAD, 0x01, STOP, FLEXSPI_1PAD, 0),
},
},
.pageDataSize = 2048,
.pageTotalSize = 4096,
.pagesPerBlock = 64,
};

```

3.6.5 Parallel NOR flash Boot over SEMC

The Smart External Memory Controller (SEMC) works in an asynchronous mode, and supports Muxed Address/Data scheme.

3.6.5.1 Parallel NOR eFUSE Configuration

The boot ROM determines the configuration of external Parallel NOR flash by parameters, either provided by eFUSE, or sampled on GPIO pins during boot. See below table for parameters details.

Table 3-25. Fuse definition for Parallel NOR over SEMC

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG [7:4]	OEM	Primary boot device selection	Yes	0000	0001 – Parallel NOR
BOOT_CFG [2:0]	OEM	SEMC Clock frequency	Yes	000	000 – 33MHz

Table continues on the next page...

Table 3-25. Fuse definition for Parallel NOR over SEMC (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					001 – 66MHz 010 – 108MHz 011 – 133MHz 1xx – 166MHz
BOOT_CFG [8]	OEM	AC timing parameter configuration	Yes	0	0 – Default configuration 1 – Configured by Fuse
BOOT_CFG [9]	OEM	Data port size	Yes	0	0 – 16 bits 1 – 8 bits
BOOT_CFG [10]	OEM	DQS pin pad enablement	Yes	0	0 – Disabled 1 – Enabled
BOOT_CFG [11]	OEM	Infinite-Loop: (Debug USE only)	Yes	0	0 - Disable 1 - Enable
0x6E0[2:0]	OEM	PCS pin selection	No	000	000 – CSX0 001 – CSX1 010 – CSX2 011 – CSX3 1xx – A8
0x6E0[5:3]	OEM	Address port size	No	000	0xx – <= 24 bits 100 – 25 bits 101 – 26 bits 110 – 27 bits 111 – 28 bits
0x6E0[6]	OEM	ADV pin polarity	No	0	0 – Low active 1 – High active
0x6E0[7]	OEM	RDY pin polarity	No	0	0 – High active 1 – Low active
0x6E0[9:8]	OEM	AC timing parameter - CES	No	00	Value = (CES*5 + 1) cycle
0x6E0[11:10]	OEM	AC timing parameter - CEH	No	00	Value = (CEH*5 + 1) cycle
0x6E0[13:12]	OEM	AC timing parameter - CEITV	No	00	Value = (CEITV*5 + 1) cycle
0x6E0[15:14]	OEM	AC timing parameter - TA	No	00	Value = (TA*5 + 1) cycle
0x6E0[19:16]	OEM	AC timing parameter - AS	No	0000	Value = (AS + 1) cycle
0x6E0[23:20]	OEM	AC timing parameter - AH	No	0000	Value = (AH + 1) cycle
0x6E0[27:24]	OEM	AC timing parameter - REL	No	0000	Value = (REL + 1) cycle

Table continues on the next page...

Table 3-25. Fuse definition for Parallel NOR over SEMC (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
0x6E0[31:28]	OEM	AC timing parameter - REH	No	0000	Value = (REH + 1) cycle

3.6.5.2 SEMC Parallel NOR Flash Boot Operation

The Boot ROM attempts to boot from Parallel NOR flash if the BOOT_CFG [7:4] fuses are programmed to 0b'0000 as shown in the Parallel NOR eFUSE Configuration table. ROM supports booting from both XIP and non-XIP images from Parallel NOR Flash. For XIP boot, the image has to be built for SEMC address space and for non XIP the image can be built to execute from Internal RAM.

3.6.6 Parallel NAND flash Boot over SEMC

The boot ROM supports a number of Parallel SLC NAND flash devices from different vendors. Both the Error Correction and Control (ECC) module in NAND device and software ECC algorithm (SECDED) in boot ROM can be used to detect the errors, based on the fuse settings.

3.6.6.1 Parallel NAND eFUSE Configuration

The boot ROM determines the configuration of external Parallel NAND flash by parameters, either provided by eFUSE, or sampled on GPIO pins, during boot. See below table for parameters details:

Table 3-26. Fuse definition for Parallel NAND over SEMC

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG [7:5]	OEM	Primary boot device selection	Yes	000	001 – Parallel NAND
BOOT_CFG [0]	OEM	Boot Search count of FCB and DBBT	Yes	0	0 - 1 1 - 2
BOOT_CFG [4:1]	OEM	Search stride for FCB and DBBT in terms of pages	Yes	0000	0000 – 64 Others – $2^{\text{BOOT_SEARCH_STRIDE}}$
BOOT_CFG [8]	OEM	NAND ONFI compliant	Yes	0	0 – ONFI 1.0

Table continues on the next page...

Table 3-26. Fuse definition for Parallel NAND over SEMC (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					1 – Non-ONFI
BOOT_CFG [9]	OEM	ECC selection	Yes	0	0 – Software ECC (SECDED) 1 – Device ECC NOTE: For “ECC selection” option, it can only be set as Device ECC When NAND device has built-in ECC module and the ECC module is enabled by default.
BOOT_CFG [10]	OEM	DQS pin pad enablement	Yes	0	0 – Disabled 1 – Enabled
BOOT_CFG [11]	OEM	Infinite-Loop: (Debug USE only)	Yes	0	0 – Disable 1 – Enable
0x6E0[2:0]	OEM	PCS pin selection	No	000	000 – CSX0 001 – CSX1 010 – CSX2 011 – CSX3 1xx – A8
0x6E0[4]	OEM	EDO mode	No	0	0 – Disabled 1 – Enabled
0x6E0[5]	OEM	RDY pin polarity	No	0	0 – High active 1 – Low active
0x6E0[6]	OEM	Ready check type	No	0	0 – R/B# pin 1 – Status Register
0x6E0[10:8]	OEM	Row Column address mode	No	000	Applicable only for Non-ONFI device 00x – 5 bytes (CA2+RA3) 010 – 4 bytes (CA2+RA2) 011 – 3 bytes (CA2+RA1) 10x – 4 bytes (CA1+RA3) 110 – 3 bytes (CA1+RA2) 111 – 2 bytes (CA1+RA1)
0x6E0[13:11]	OEM	Column address width	No	000	Applicable only for Non-ONFI device 000 – 12 bits 001 – 09 bits 010 – 10 bits 011 – 11 bits

Table continues on the next page...

Table 3-26. Fuse definition for Parallel NAND over SEMC (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					100 – 13 bits 101 – 14 bits 110 – 15 bits 111 – 16 bits
0x6E0[14]	OEM	Status command type	No	0	Applicable only for Non-ONFI device 0 – Common (0x70) 1 – Enhanced (0x78)
0x6E0[18:16]	OEM	Pages in block	No	000	Applicable only for Non-ONFI device 000 – 128 pages 001 – 8 pages 010 – 16 pages 011 – 32 pages 100 – 64 pages 101 – 256 pages 110 – 512 pages 111 – 1024 pages
0x6E0[24]	OEM	Device ECC initial status	No	0	Applicable only for ONFI 1.0 device 0 – Enabled 1 – Disabled

3.6.6.2 Parallel NAND Flash Boot Control Blocks (BCB)

There are two BCB data structures:

- Firmware Configuration Block (FCB)
- Discovered Bad Blocks Table (DBBT)

As part of the Parallel NAND media initialization, the ROM driver uses proper Parallel NAND parameters specified by FUSE to search for an FCB that contains the complete Parallel NAND parameters, page address of DBBT Search Area and Image info, including image copies, start page address and image size in terms of pages for each image.

FCB data structure is protected using Embedded ECC module in Parallel NAND devices or software ECC in ROM. The ROM driver reads 2048 bytes of first sector and checks the ECC check status to determine whether FCB data is valid or not.

Boot devices (internal boot)

If the FCB is found, the complete NAND parameters (Parallel NAND configuration block) are loaded for further reads, if the ECC fails, or the fingerprint does not match, or the CRC checksum does not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next FCB until Search Count pages have been read.

If search fails to find a valid FCB, the Parallel NAND driver responds with an error and the boot ROM enters into Recovery boot mode (Secondary boot, if it is enabled, or Serial download mode).

The FCB contains the page address of DDBT Search Area, and the info for images. DDBT is searched in DDBT Search area just like how FCB is searched. After the FCB is read, the DDBT is loaded, then the boot image is loaded using starting page address from FCB.

3.6.6.3 Firmware Configuration Block (FCB)

The FCB is at the first page in the first good block. The FCB should be present at each search stride of the search area.

The search area contains copies of the FCB at each stride distance, in case the first Serial NAND block becomes corrupted, the ROM will find its copy in the next Parallel NAND block. The search area should span over at least two Parallel NAND blocks. The location information for DDBT search area and images are all specified in the FCB. Below Table shows the FCB Structure.

Name	offset	Size (Bytes)	Description
bcbHeader	0x000	12	See Table 3-27 for details
DDBTSerachAreaStartPage	0x00c	4	Start Page address for bad block table search area
searchStride	0x010	2	Search stride for DDBT and FCB search. Not used by ROM Max value is 8.
searchCount	0x012	2	Copies of DDBT and FCB. Not used by ROM, max value is 8.
firmwareCopies	0x014	4	Firmware copies Valid range 1-8.
-	0x018	40	Reserved
firmwareTable	0x040	64	For details see Table 3-28
-	0x080	128	Reserved
nandConfig	0x100	256	Parallel NAND configuration block over SEMC
-	0x200	512	Reserved

Table 3-27. Header Description

Field	Size	Description
crcChecksum	4	Checksum
fingerprint	4	0x4E46_4342 ASCII: “NFCB”
version	4	0x0000_0001

Table 3-28. Table Descriptions

Field	Size	Description
startPage	4	Start page of this firmware
pagesInFirmware	4	Pages in this firmware

NOTE

- The “crcChecksum” is calculated with an MPEG2 variant of CRC-32.
- The “crcChecksum” calculation starts from fingerprint to the end of FCB, 1020 bytes in total.
- The “nandConfig” is SEMC NAND configuration block which consists of common SEMC memory configuration block and Parallel NAND specified configuration parameters. See [Parallel NAND eFUSE Configuration](#) for more details.

3.6.6.4 Discovered Bad Blocks Table (DBBT)**Table 3-29. DBBT Structure**

Name	offset	Size (Bytes)	Description
bcbHeader	0x000	12	See Table 3-30 for details
-	0x00c	4	Reserved
badBlockNumber	0x010	4	Number of bad blocks
-	0x014	12	Reserved
badBlockTable	0x020	1024	Bad Block entries, only the first “badBlockNumber” entries are valid, the remaining entries must be filled with 0x00s

Table 3-30. Header Description

Field	Size	Description
<i>Table continues on the next page...</i>		

Table continues on the next page...

Table 3-30. Header Description (continued)

crcChecksum	4	Checksum
fingerprint	4	0x4442_4254 ASCII: "DBBT"
version	4	0x0000_0001

NOTE

- Maximum bad block number is 256.
- The "crcChecksum" is calculated with the same algorithm as the one in FCB, from Fingerprint to the end of DBBT, 1052 bytes in total.

3.6.6.5 Bad block handling in ROM

During firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block.

If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block is checked.

3.6.7 Parallel NOR and NAND configuration based on SEMC interface

The ROM SW supports Parallel NOR and Parallel NAND based on SEMC module, using an 80-bytes common SEMC configuration block and several specified parameters for Parallel NOR and Parallel NAND respectively. See below sections for more details.

3.6.7.1 SEMC Configuration Block

SEMC Configuration block consists of all parameters related to specific Flash devices.

Table 3-31. SEMC control block structure

Name	offset	Size (Bytes)	Description
tag	0x000	4	0x434D4553, ascii:"SEMC"
version	0x004	4	0x00010000 [07:00] bugfix = 0 [15:08] minor = 0 [31:16] major = 1

Table continues on the next page...

Table 3-31. SEMC control block structure (continued)

Name	offset	Size (Bytes)	Description
deviceMemType	0x008	1	0 – NOR Flash 1 – NAND Flash
accessCommandType	0x009	1	0 – IPG bus command 1 – AXI32 command
-	0x00A	2	Reserved
asyncClkFreq	0x00C	1	0 – 33MHz 1 – 40MHz 2 – 50MHz 3 – 66MHz 4 – 108MHz 5 – 133MHz 6 – 166MHz
busTimeoutCycles	0x00D	1	0 – 255 * 1024 cycles n – n * 1024 cycles
commandExecutionTimeoutCycles	0x00E	1	0 – 256 * 1024 cycles n – n * 1024 cycles
readStrobeMode	0x00F	1	0 – Dummy read strobe loopbacked internally 1 – Dummy read strobe loopbacked from DQS pad
norMemConfig	0x010	64	See detail in Table SEMC NOR control block structure
nandMemConfig	0x010	64	See detail in Table SEMC NAND control block structure

Table 3-32. SEMC NOR control block structure

Name	offset	Size (Bytes)	Description
comMemBaseAddress	0x00	4	SoC level Base Address for NAND AXI and IPG command
comMemSizeInByte	0x04	4	SoC level Memory size for NAND AXI and IPG command
-	0x08	8	Reserved
addressMode	0x10	1	0 – Address/Data MUX mode 1 – Advanced Address/Data MUX mode 2 – Address/Data non-MUX mode
addressPortWidth	0x11	1	Address Port bit number
dataPortWidth	0x12	1	Data Port bit number
columnAddressWidth	0x13	1	Column Address bit width

Table continues on the next page...

Table 3-32. SEMC NOR control block structure (continued)

Name	offset	Size (Bytes)	Description
burstLengthInBytes	0x14	1	Burst Length
	0x15	3	Reserved
cePortOutputSelection	0x18	1	0 – CSX0 1 – CSX1 2 – CSX2 3 – CSX3 4 – A8
rdyPortPolarity	0x19	1	0 – Low active 1 – High active
advPortPolarity	0x1A	1	0 – Low active 1 – High active
-	0x1B	13	Reserved
ceSetupTime	0x28	1	value[3:0] + 1 cycles
ceMinHoldTime	0x29	1	value[3:0] + 1 cycles
ceMinIntervalTime	0x2A	1	value[3:0] + 1 cycles
addressSetupTime	0x2B	1	value[3:0] + 1 cycles
addressHoldTime	0x2C	1	value[3:0] + 1 cycles
asyncWeLowTime	0x2D	1	value[3:0] + 1 cycles
asyncWeHighTime	0x2E	1	value[3:0] + 1 cycles
asyncOeLowTime	0x2F	1	value[3:0] + 1 cycles
asyncOeHighTime	0x30	1	value[3:0] + 1 cycles
asyncTurnaroundTime	0x31	1	value[3:0] + 1 cycles
asyncAddressToDataHoldTime	0x32	1	value[3:0] + 1 cycles
syncDataSetupTime	0x33	1	value[3:0] + 1 cycles
syncDataHoldTime	0x34	1	value[3:0] + 1 cycles
syncLatencyCount	0x35	1	value[3:0] + 1 cycles
syncReadCycleTime	0x36	1	value[3:0] + 1 cycles
-	0x37	9	Reserved

Table 3-33. SEMC NAND control block structure

Name	offset	Size (Bytes)	Description
Name	offset	Size (Bytes)	Description
axiMemBaseAddress	0x00	4	SoC level Base Address for NAND AXI command
axiMemSizeInByte	0x04	4	SoC level Memory size for NAND AXI command
ipgMemBaseAddress	0x08	4	SoC level Base Address for NAND IPG command

Table continues on the next page...

Table 3-33. SEMC NAND control block structure (continued)

Name	offset	Size (Bytes)	Description
ipgMemSizeInByte	0x0c	4	SoC level Memory size for NAND IPG command
edoMode	0x10	1	0 - EDO mode disabled 1 - EDO mode enabled
ioPortWidth	0x11	1	IO Port bit number
arrayAddressOption	0x12	1	0 – 5 bytes (CA2+RA3) 1 – 4 bytes (CA1+RA3) 2 – 4 bytes (CA2+RA2) 3 – 3 bytes (CA1+RA2) 4 – 3 bytes (CA2+RA1) 7 – 2 bytes (CA1+RA1)
columnAddressWidth	0x13	1	Column address bit number
burstLengthInBytes	0x14	1	Burst Length
-	0x15	11	Reserved
cePortOutputSelection	0x20	1	0 – CSX0 1 – CSX1 2 – CSX2 3 – CSX3 4 – A8
rdyPortPolarity	0x21	1	0 – Low active 1 – High active
-	0x22	14	Reserved
ceSetupTime	0x30	1	value[3:0] + 1 cycles
ceMinHoldTime	0x31	1	value[3:0] + 1 cycles
ceMinIntervalTime	0x32	1	value[3:0] + 1 cycles
weLowTime	0x33	1	value[3:0] + 1 cycles
weHighTime	0x34	1	value[3:0] + 1 cycles
reLowTime	0x35	1	value[3:0] + 1 cycles
reHighTime	0x36	1	value[3:0] + 1 cycles
weHighToReLowTime	0x37	1	value[5:0] + 1 cycles
reHighToWeLowTime	0x38	1	value[5:0] + 1 cycles
aleToDataStartTime	0x39	1	value[5:0] + 1 cycles
readyToReLowTime	0x3a	1	value[5:0] + 1 cycles
weHighToBusyTime	0x3b	1	value[5:0] + 1 cycles
asyncTurnaroundTime	0x3c	1	value[3:0] + 1 cycles
-	0x3d	3	Reserved

3.6.7.2 Parallel NOR Configuration Block (80 bytes)

Table 3-34. Parallel NOR control block structure

Name	offset	Size (Bytes)	Description
memConfig	0x000	80	See SEMC control block structure for more details

3.6.7.3 Parallel NAND Configuration Block (256 bytes)

Table 3-35. Parallel NAND control block structure

Name	offset	Size (Bytes)	Description
memConfig	0x000	80	See SEMC control block structure for more details
vendorType	0x050	1	0 – Micron 1 – Spansion 2 – Samsung 3 – Winbond 4 – Hynix 5 – Toshiba 6 – Macronix
cellTechnology	0x051	1	0 – SLC 1 – MLC
onfiVersion	0x052	1	0 – Non-ONFI 1 – ONFI 1.0 2 – ONFI 2.0 3 – ONFI 3.0 4 – ONFI 4.0
acTimingTableIndex	0x053	1	0 – User Defined 1 – ONFI 1.0 Mode0 10MHz 2 – ONFI 1.0 Mode1 20MHz 3 – ONFI 1.0 Mode2 28MHz 4 – ONFI 1.0 Mode3 33MHz 5 – ONFI 1.0 Mode4 40MHz 6 – ONFI 1.0 Mode5 50MHz 7 – Auto Detection
enableEccCheck	0x054	1	0 – Enabled 1 – Disabled
eccCheckType	0x055	1	0 – Software ECC 1 – Device ECC
deviceEccStatus	0x056	1	0 – Enabled

Table continues on the next page...

Table 3-35. Parallel NAND control block structure (continued)

Name	offset	Size (Bytes)	Description
			1 – Disabled
swEccAlgorithm	0x057	1	0 – SEC Hamming Code
swEccBlockBytes	0x058	4	Software ECC block bytes (256, 512)
readyCheckOption	0x05c	1	0 – Via Status Register 1 – Via R/B# signal
statusCommandType	0x05d	1	0 – Common (0x70) 1 – Enhanced (0x78)
readyCheckTimeoutInMs	0x05e	2	Ready Check timeout
readyCheckIntervalInUs	0x060	2	Ready Check interval
-	0x062	62	Reserved
bytesInPageDataArea	0x0a0	4	Page Main data size
bytesInPageSpareArea	0x0a4	4	Page Spare data size
pagesInBlock	0x0a8	4	Page number in one block
blocksInPlane	0x0ac	4	Block number in one plane
planesInDevice	0x0b0	4	Plane number in Device
-	0x0b4	76	Reserved

3.6.8 Expansion device

The ROM supports booting from the MMC/eMMC and SD/eSD compliant devices.

3.6.8.1 Expansion device eFUSE configuration

The SD/MMC/eSD/eMMC/SDXC boot can be performed using the USDHC ports. The port can be configured based on either the setting of the BOOT_CFG1[1] (Port Select) fuse or its corresponding GPIO overrides.

All USDHC ports support the fast boot. See this table for details:

Table 3-36. USDHC eFuse Descriptions

Fuse	Config	Definition	GPIO	Shipped value	Settings
0x450[7:6]	OEM	Boot device selection	Yes	00	01 - SD/eSD/SDXC boot from the USDHC interface 10 - MMC/eMMC boot from the USDHC interface

Table continues on the next page...

Table 3-36. USDHC eFuse Descriptions (continued)

0x450[5:4]	OEM	SD/MMC speed mode, and eMMC acknowledge enabled selection	Yes	00	MMC 0x - Normal speed mode 1x - High-speed mode x0 - eMMC fast boot acknowledge disable x1 - eMMC fast boot acknowledge enable SD 00 - Normal/SDR12 01 - High/SDR25 10 - SDR50 11 - SDR104
0x450[3]	OEM	SD power cycle enable	Yes	0	0 - No power cycle 1 - Power cycle enabled via the SD_RST pad
0x450[2]	OEM	SD loopback clock source sel (for SDR50 and SDR104 only)	Yes	0	0 - through the SD pad 1 - direct
0x450[1]	OEM	USDHC port selection	Yes	0	0 - USDHC-1 1 - USDHC-2
0x450[0]	OEM	Fast boot support	Yes	0	MMC 0 - Normal boot 1 - Fast boot
0x450[10:9]	OEM	SD MMC bus width selection	Yes	00	SD x0 - 1-bit x1 - 4-bit MMC 00 - 4-bit 01 - 8-bit 10 - 4-bit DDR (MMC 4.4) 11 - 8-bit DDR (MMC 4.4)
0x450[8]	OEM	USDHC1 voltage selection	Yes	0	0 - 3.3 V 1 - 1.8 V
0x460[31:30]	OEM	Power cycle selection	Yes	00	00 - 20 ms 01 - 10 ms 10 - 5 ms 11 - 2.5 ms
0x460[29]	OEM	Power stable cycle selection	Yes	0	0 - 5 ms 1 - 2.5 ms
0x460[24]	OEM	SD/MMC DLL enable selection	Yes	0	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/Emm
0x470[7]	OEM	DLL override selection	Yes	0	0 - No override 1 - DLL override mode for SD/eMMC (Override by MMC_DLL_DLY, 0x470[19:16])

Table continues on the next page...

Table 3-36. USDHC eFuse Descriptions (continued)

0X470[6]	OEM	USDHC1 reset polarity selection	Yes	0	0 - Reset active low 1 - Reset active high
0X470[5]	OEM	USDHC2 voltage selection	Yes	0	0 - 3.3 V 1 - 1.8 V
0X470[3]	OEM	Disable SDMMC Manufacture mode	Yes	0	0 - Enable 1 - Disable
0X470[0]	OEM	SD/MMC pad settings override selection	Yes	0	0 - No override 1 - Override (override by PAD_SETTINGS, 0x6d0[5:0])
0X470[15]	OEM	USDHC2 reset polarity selection	Yes	0	0 - Reset active-low 1 - Reset active-high
0X470[14]	OEM	eMMC4.4 pre-idle enabled selection	Yes	0	0 - Issue pre-idle command 1 - Do not issue
0x470[13]	OEM	Override HYS bit for SD/MMC pads	Yes	0	0 - No override 1 - Override HYS bit with 1
0x470[12]	OEM	USDHC_PAD_PULL_D OWN	Yes	0	SD 0 - no action 1 - pull down
0x470[11]	OEM	ENABLE_EMMC_22K_PULLUP	Yes	0	MMC 0 - 47K pullup 1 - 22K pullup
0x470[9]	OEM	USDHC_IOMUX_SION_BIT_ENABLE	Yes	0	0 - Disable 1 - Enable
0x470[8]	OEM	USDHC_IOMUX_SRE_ENABLE	Yes	0	0 - Disable 1 - Enable
0X470[30:24]	OEM	MMC_DLL_DLY	Yes	0000000	Override number
0X6D0[5:0]	OEM	PAD_SETTINGS	Yes	000000	Override number
0X6D0[31:30]	OEM	SD calibration step	Yes	00	SD 00 - 1 delay cell 01 - 2 delay cells 10 - 4 delay cells 11 - 4 delay cells

The boot code supports these standards:

- MMCv4.4 or less
- eMMCv4.4 or less
- SDv2.0 or less
- eSDv2.10 rev-0.9, with or without FAST_BOOT
- SDXCv3.0

Boot devices (internal boot)

The MMC/SD/eSD/SDXC/eMMC can be connected to any of the USDHC blocks and can be booted by copying 4 KB of data from the MMC/SD/eSD/eMMC device to the internal RAM. After checking the Image Vector Table header value (0xD1) from program image, the ROM code performs a DCD check. After a successful DCD extraction, the ROM code extracts from the Boot Data Structure the destination pointer and length of image to be copied to the RAM device from where the code execution occurs.

The maximum image size to load into the SD/MMC boot is 32 MB. This is due to a limited number of uSDHC ADMA Buffer Descriptors allocated by the ROM.

NOTE

The initial 4 KB of the program image must contain the IVT, DCD, and the Boot Data structures.

Table 3-37. SD/MMC frequencies

	SD	MMC	MMC (DDR mode)
Identification (KHz)	347.22		
Normal-speed mode (MHz)	25	20	25
High-speed mode (MHz)	50	40	50
UHSI SDR50 (MHz)	100		
UHSI SDR104 (MHz)	200		

NOTE

The boot ROM code reads the application image length and the application destination pointer from the image.

3.6.8.2 MMC and eMMC boot

This table provides the MMC and eMMC boot details.

Table 3-38. MMC and eMMC boot details

Normal boot mode	During the initialization (normal boot mode), the MMC frequency is set to 347.22 KHz. When the MMC card enters the identification portion of the initialization, the voltage validation is performed, and the ROM boot code checks the high-voltage settings and the card capacity. The ROM boot code supports both the high-capacity and low-capacity MMC/eMMC cards. After the initialization phase is complete, the ROM boot code switches to a higher frequency (20 MHz in the normal boot mode or 40 MHz in the high-speed mode). The eMMC is also interfaced via the USDHC and follows the same flow as the MMC.
------------------	--

Table continues on the next page...

Table 3-38. MMC and eMMC boot details (continued)

	The boot partition can be selected for an MMC4.x card after the card initialization is complete. The ROM code reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in the BOOT_PARTITION_ENABLE field or the user partition was mentioned, the ROM boots from the user partition.
eMMC4.3 or eMMC4.4 device supporting special boot mode	If using an eMMC4.3 or eMMC4.4 device that supports the special boot mode, it can be initiated by pulling the CMD line low. If the BOOT ACK is enabled, the eMMC4.3/eMMC4.4 device sends the BOOT ACK via the DATA lines and the ROM can read the BOOT ACK [S010E] to identify the eMMC4.3/eMMC4.4 device. The eMMC4.3/eMMC4.4 device with the "boot mode" feature can only be supported via the ESDHCV3-3 and with or without the BOOT ACK. If the BOOT ACK is enabled, the ROM waits 50 ms to get the BOOT ACK and if the BOOT ACK is received by the ROM. If BOOT ACK is disabled ROM waits 1 second for data. If the BOOT ACK or data was received, the eMMC4.3/eMMC4.4 is booted in the "boot mode", otherwise the eMMC4.3/eMMC4.4 boots as a normal MMC card from the selected boot partition. This boot mode can be selected by the BOOT_CFG1[4] (fast boot) fuse. The BOOT ACK is selected by the BOOT_CFG2[1].
eMMC4.4 device	If using the eMMC4.4 device, the Double Data Rate (DDR) mode can be used. This mode can be selected by the BOOT_CFG2[7:5] (bus width) fuse.

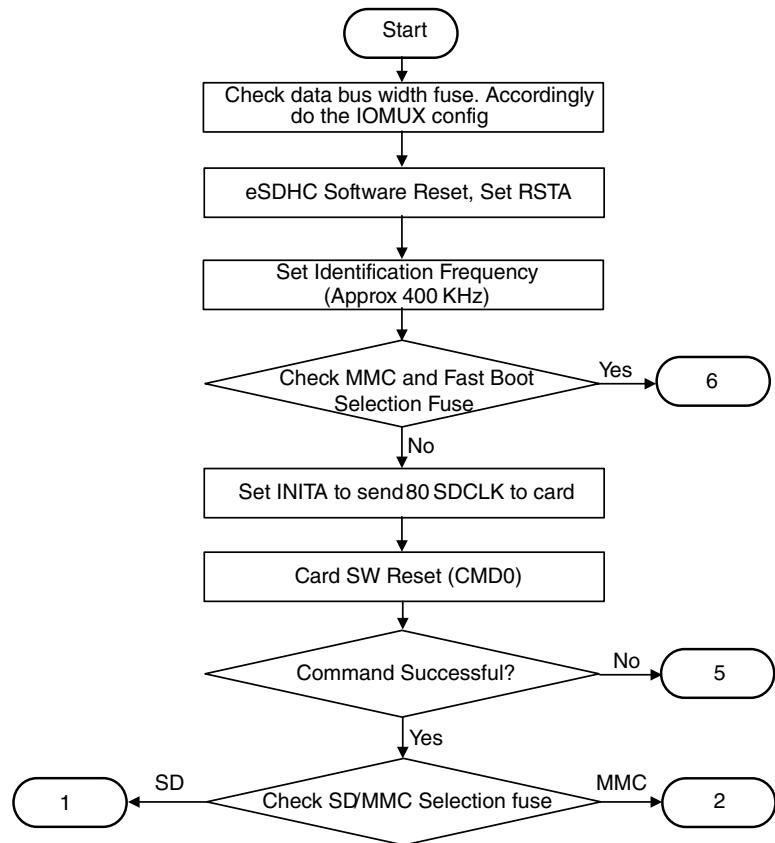


Figure 3-7. Expansion device boot flow (1 of 6)

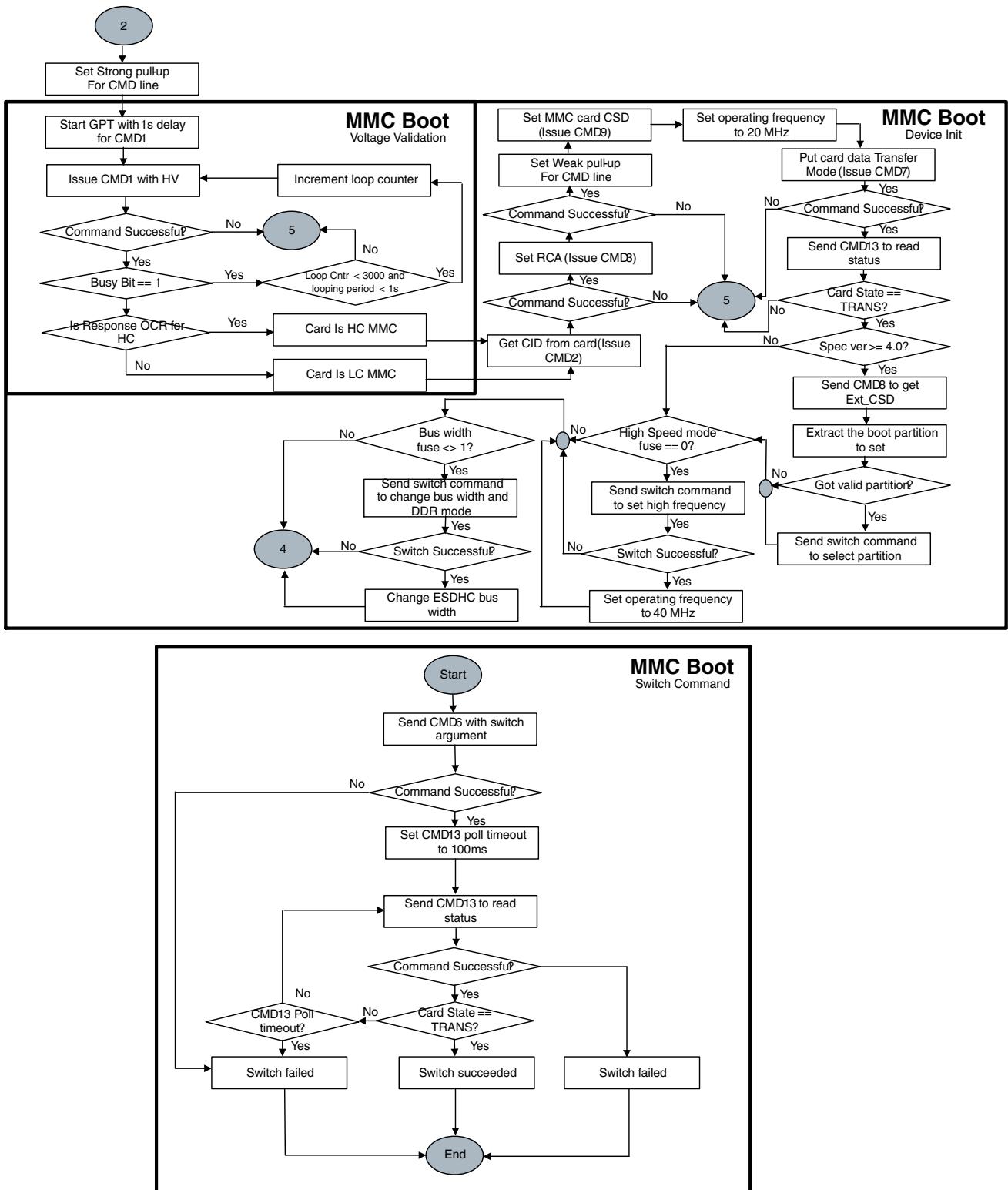


Figure 3-8. Expansion device (MMC) boot flow (2 of 6)

Boot devices (internal boot)

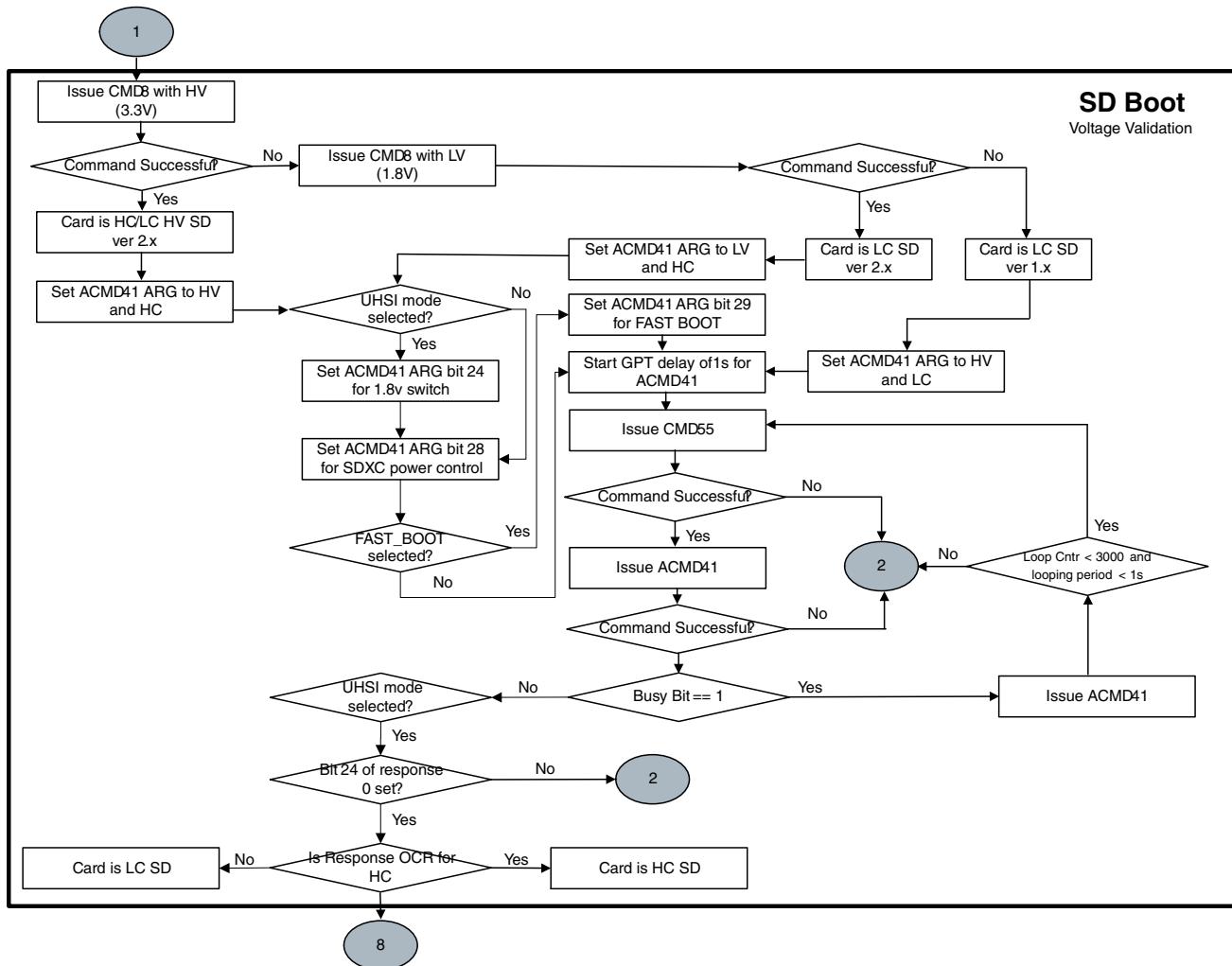


Figure 3-9. Expansion device (SD/eSD/SDXC) boot flow (3 of 6) part 1

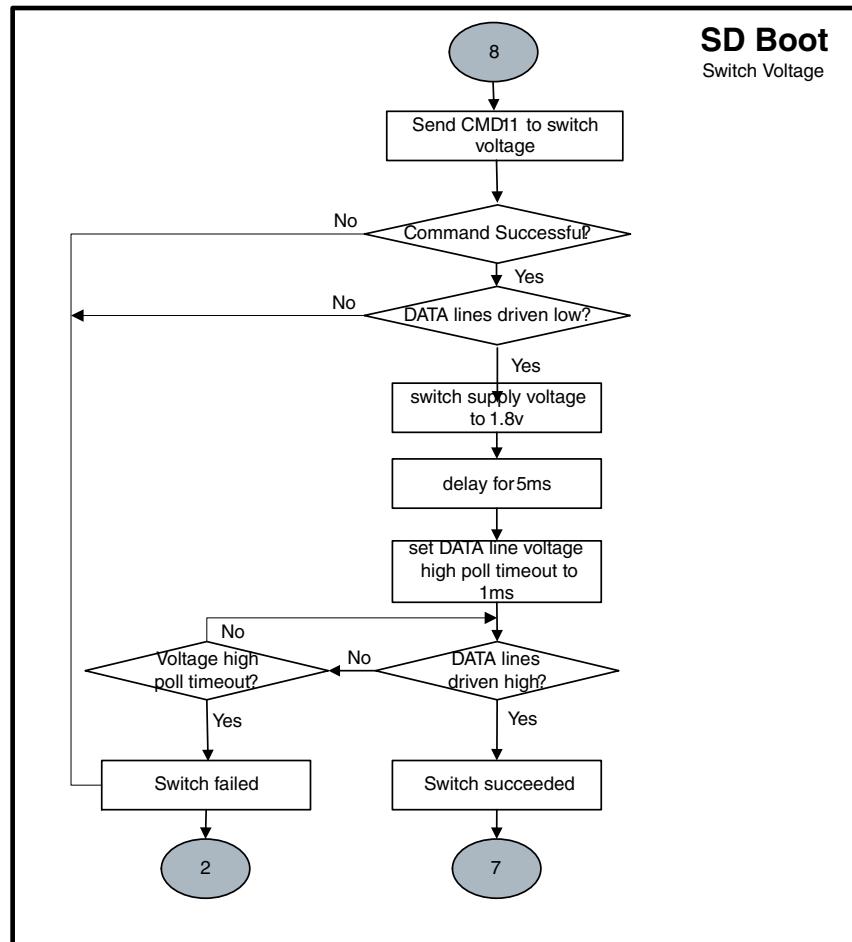


Figure 3-10. Expansion device (SD/eSD/SDXC) boot flow (3 of 6) part 2

Boot devices (internal boot)

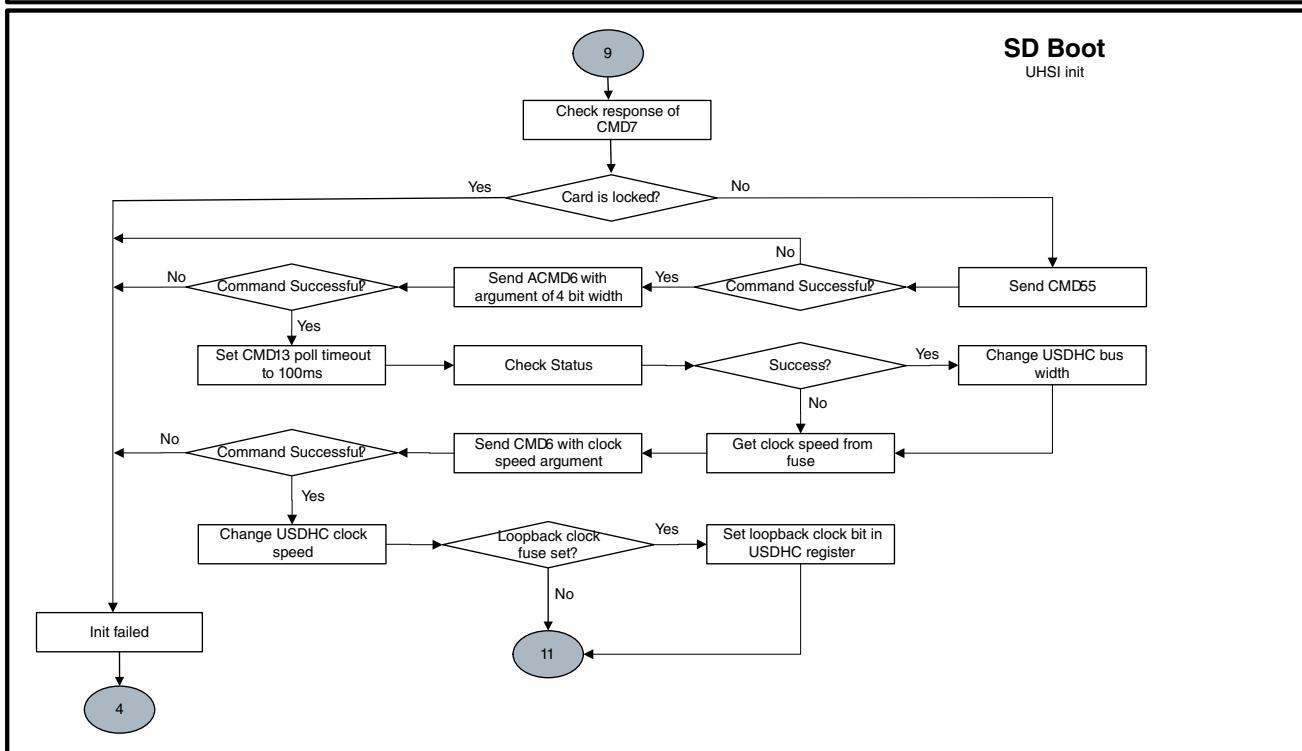
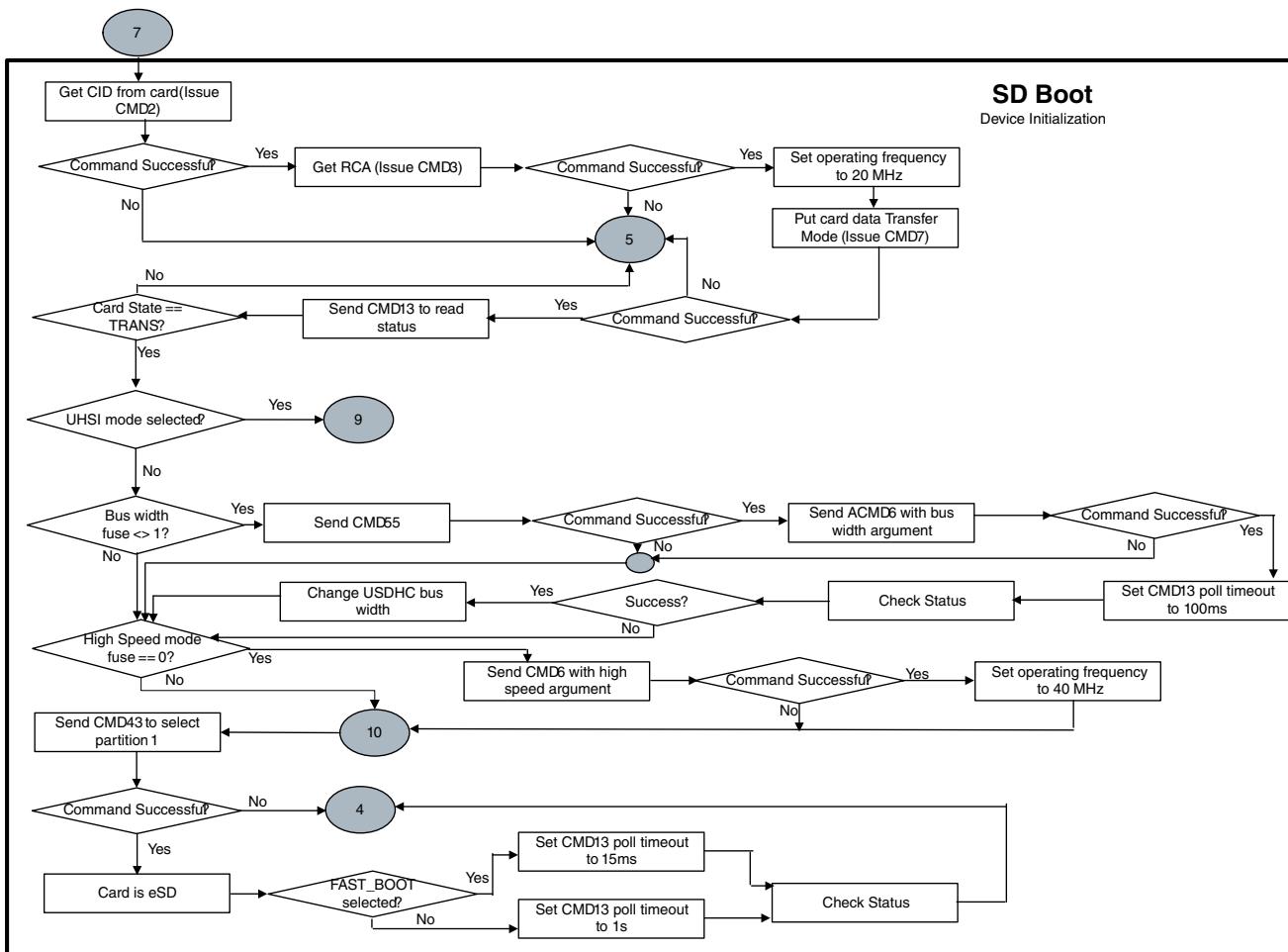


Figure 3-11. Expansion device (MMCSD/eSD/SDXC) boot flow (4 of 6)
Security Reference Manual for the i.MX RT1050 Processor, Rev. 1, 04/2018

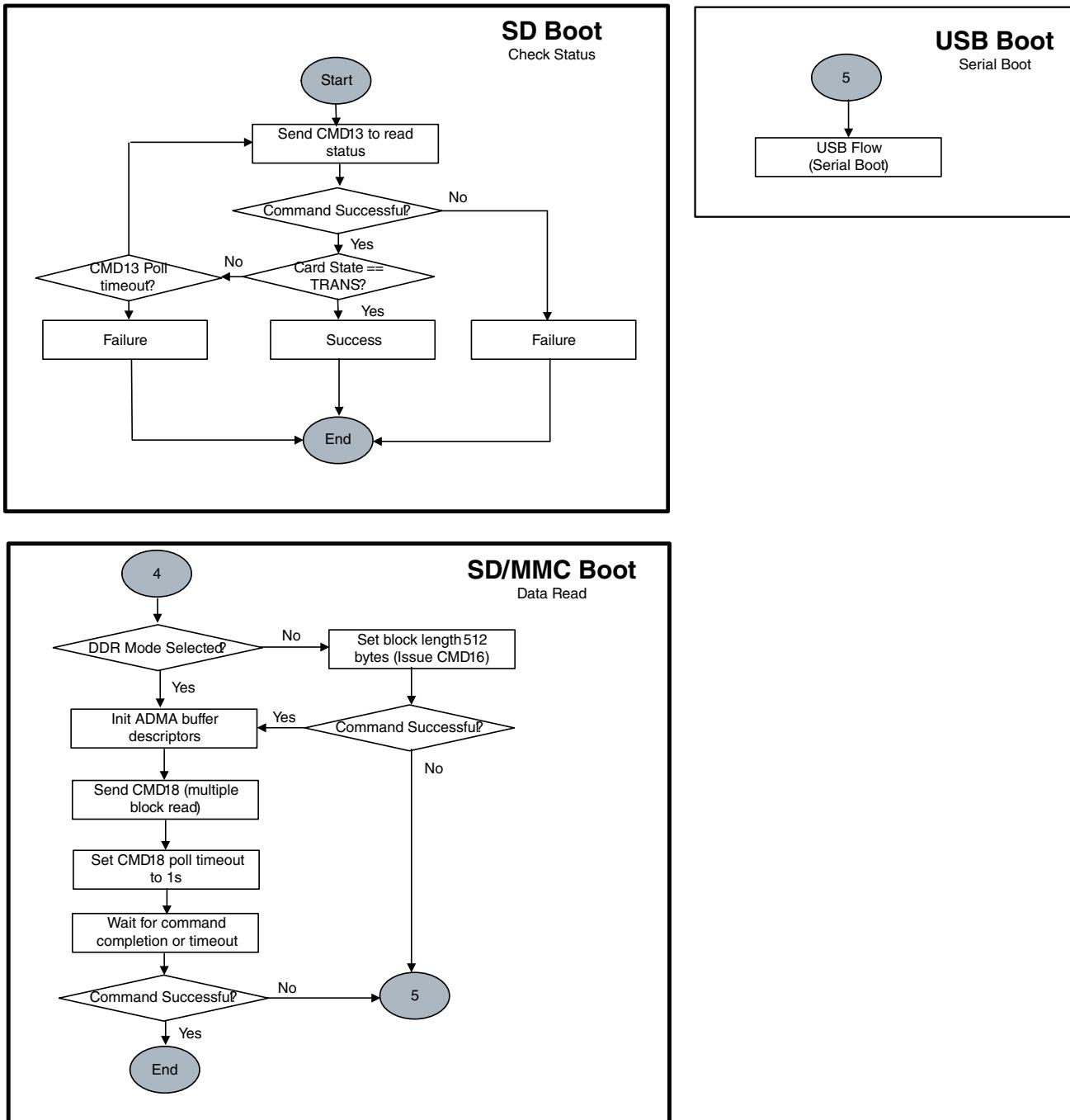


Figure 3-12. Expansion device (SD/eSD) boot flow (5 of 6)

Boot devices (internal boot)

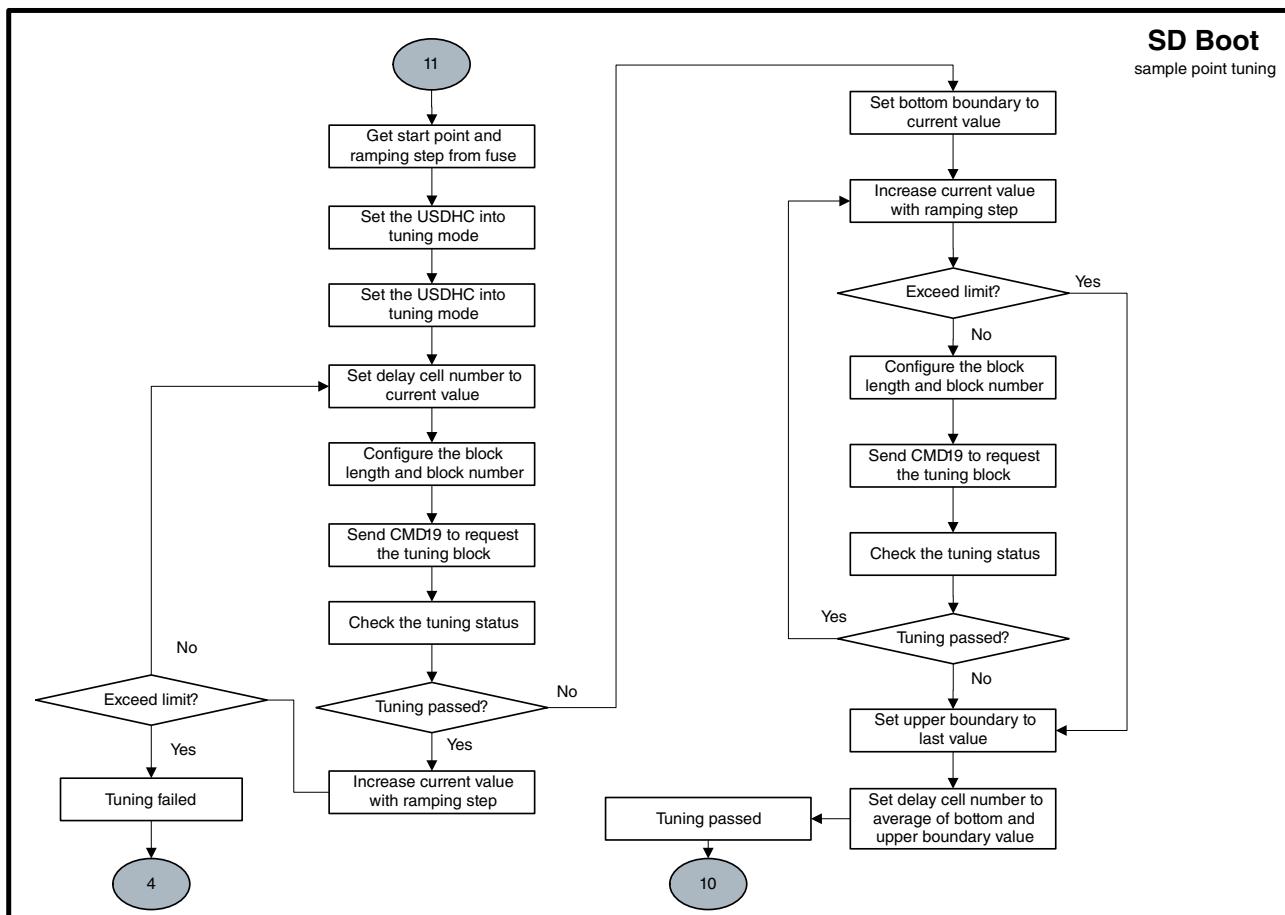
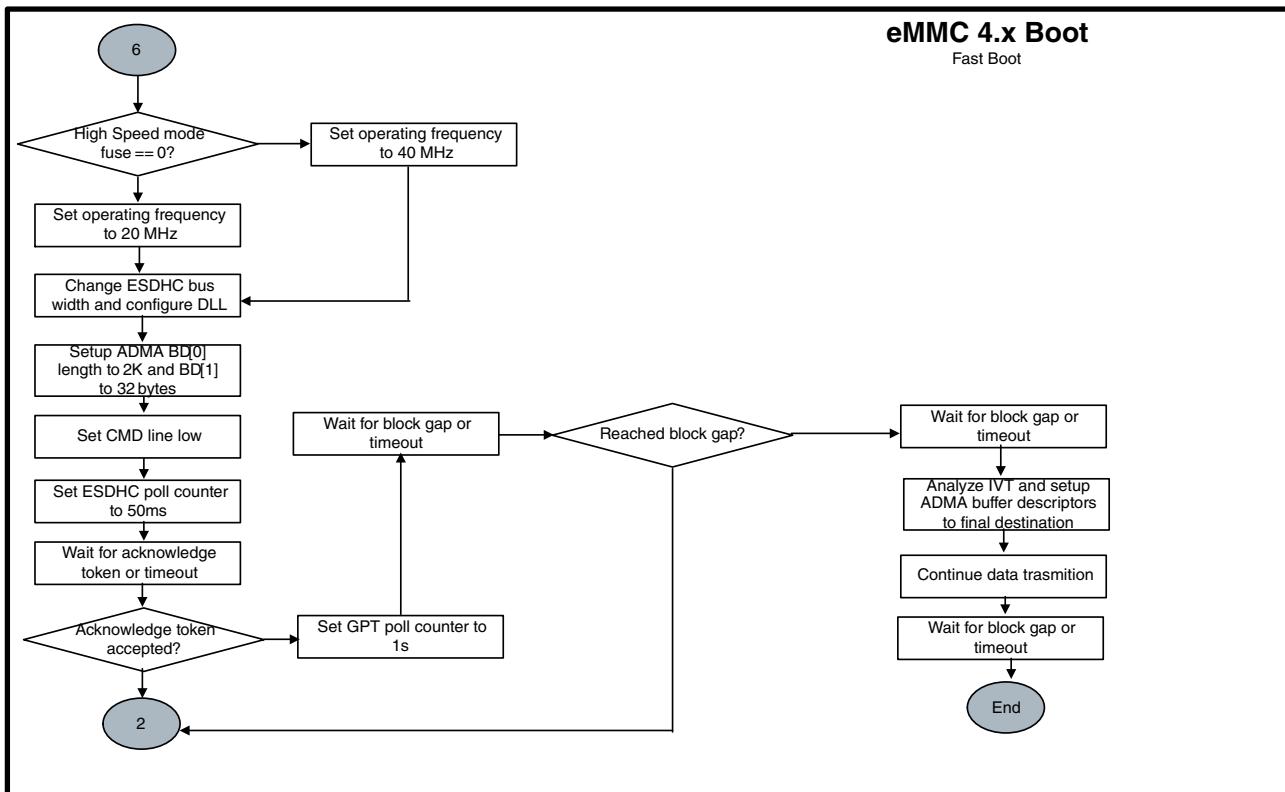


Figure 3-13. Expansion device boot flow (6 of 6)
Security Reference Manual for the i.MX RT1050 Processor, Rev. 1, 04/2018

3.6.8.3 SD, eSD, and SDXC

After the normal boot mode initialization begins, the SD/eSD/SDXC frequency is set to 347.22 kHz. During the identification phase, the SD/eSD/SDXC card voltage validation is performed. During the voltage validation, the boot code first checks with the high-voltage settings; if that fails, it checks with the low-voltage settings.

The capacity of the card is also checked. The boot code supports the high-capacity and low-capacity SD/eSD/SDXC cards after the voltage validation card initialization is done.

During the card initialization, the ROM boot code attempts to set the boot partition for all SD, eSD, and SDXC devices. If this fails, the boot code assumes that the card is a normal SD or SDXC card. If it does not fail, the boot code assumes it is an eSD card. After the initialization phase is over, the boot code switches to a higher frequency (25 MHz in the normal-speed mode or 50 MHz in the high-speed mode). The ROM also supports the FAST_BOOT mode booting from the eSD card. This mode can be selected by the BOOT_CFG1[0] (Fast Boot).

For the UHSI cards, the clock speed fuses can be set to SDR50 or SDR104 on USDHC1, USDHC2 ports. This enables the voltage switch process to set the signaling voltage to 1.8 V during the voltage validation. The bus width is fixed at a 4-bit width and a sampling point tuning process is needed to calibrate the number of the delay cells. If the SD Loopback Clock eFuse is set, the feedback clock comes directly from the loopback SD clock, instead of the card clock (by default). The SD clock speed can be selected by the BOOT_CFG1[5:4], and the SD Loopback Clock is selected by the BOOT_CFG1[2].

The UHSI calibration start value (MMC_DLL_DL_Y[6:0]) and the step value SD_CALIBRATION_STEP[1:0] can be set to optimize the sample point tuning process.

If the SD Power Cycle Enable eFuse is 1, the ROM sets the SD_RST pad low, waits for 5 ms, and then sets the SD_RST pad high. If the SD_RST pad is connected to the SD power supply enable logic on board, it enables the power cycle of the SD card. This may be crucial in case the SD logic is in the 1.8 V states and must be reset to the 3.3 V states.

The SDR50 and SDR104 boots are not supported on the USDHC1 and USDHC2 ports because there are no reset signals for those ports when connected in the IOMUX.

3.6.8.4 IOMUX configuration for SD/MMC

Table 3-39. IOMUX configuration for SD/MMC

Signal	USDHC1	USDHC2
CLK	GPIO_SD_B0_01.alt0	GPIO_SD_B1_04.alt0
CMD	GPIO_SD_B0_00.alt0	GPIO_SD_B1_05.alt0
DATA0	GPIO_SD_B0_02.alt0	GPIO_SD_B1_03.alt0
DATA1	GPIO_SD_B0_03.alt0	GPIO_SD_B1_02.alt0
DATA2	GPIO_SD_B0_04.alt0	GPIO_SD_B1_01.alt0
DATA3	GPIO_SD_B0_05.alt0	GPIO_SD_B1_00.alt0
DATA4	-	GPIO_SD_B1_08.alt0
DATA5	-	GPIO_SD_B1_09.alt0
DATA6	-	GPIO_SD_B1_10.alt0
DATA7	-	GPIO_SD_B1_11.alt0
VSELECT	GPIO_B1_14.alt6	GPIO_EMC_38.alt6
RESET_B	GPIO_B1_15.alt6	GPIO_SD_B1_06.alt0
CD_B	GPIO_B1_12.alt6	-

3.6.8.5 Redundant boot support for expansion device

The ROM supports the redundant boot for an expansion device. The primary or secondary image is selected, depending on the PERSIST_SECONDARY_BOOT setting. (see [Table 3-7](#)).

If the PERSIST_SECONDARY_BOOT is 0, the boot ROM uses address 0x0 for the primary image.

If the PERSIST_SECONDARY_BOOT is 1, the boot ROM reads the secondary image table from address 0x200 on the boot media and uses the address specified in the table.

Table 3-40. Secondary image table format

Reserved (chipNum)
Reserved (driveType)
tag
firstSectorNumber
Reserved (sectorCount)

Where:

- The tag is used as an indication of the valid secondary image table. It must be 0x00112233.
- The firstSectorNumber is the first 512-byte sector number of the secondary image.

For the secondary image support, the primary image must reserve the space for the secondary image table. See this figure for the typical structures layout on an expansion device.

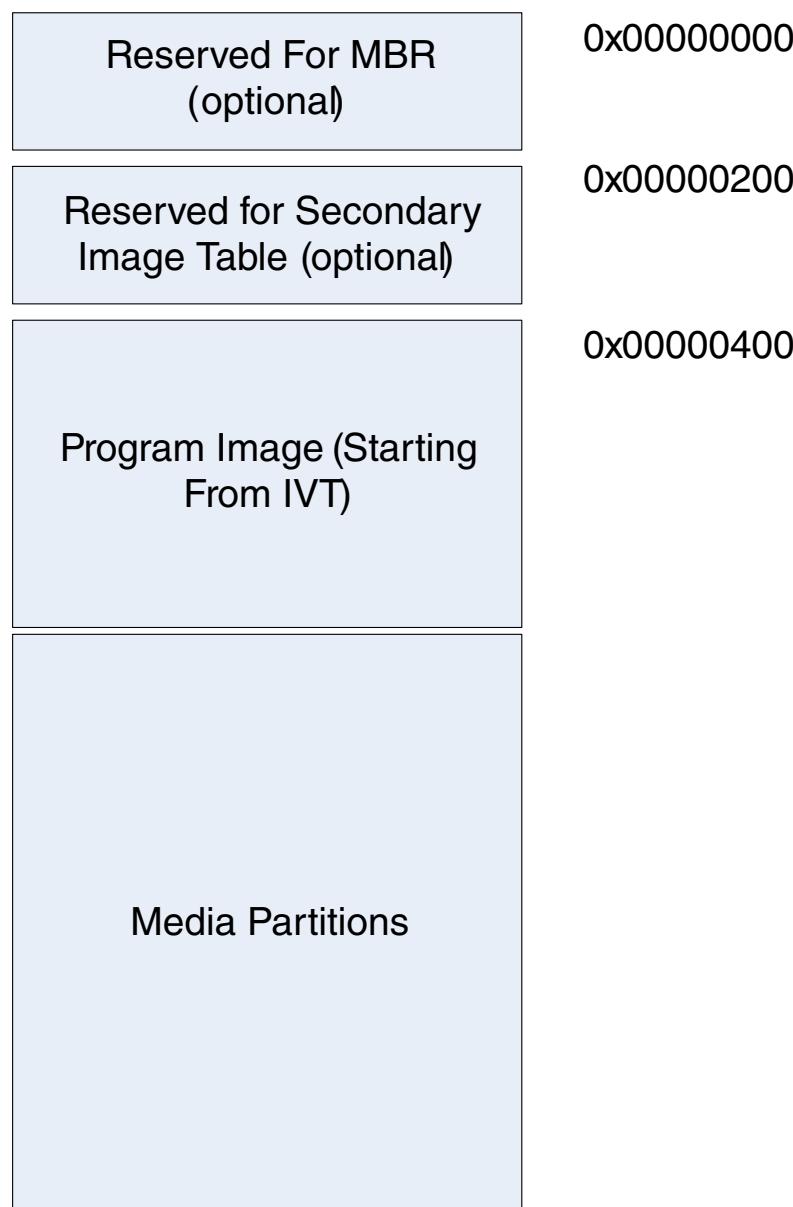


Figure 3-14. Expansion device structures layout

For the Closed mode, if there are failures during primary image authentication, the boot ROM turns on the PERSIST_SECONDARY_BOOT bit (see [Table 3-7](#)) and performs the software reset. (After the software reset, the secondary image is used.)

3.6.9 Serial NOR/EEPROM through LPSPI

The chip supports booting from serial memory devices, such as EEPROM and serial flash, using the LPSPI.

These ports are available for serial boot: LPSPI (LPSPI1, LPSPI2, LPSPI3, LPSPI4) interfaces.

3.6.9.1 Serial NOR/EEPROM eFUSE configuration

The boot ROM code determines the type of device using the following parameters, either provided by the eFUSE settings or sampled on the I/O pins, during boot.

Table 3-41. Serial NOR/EEPROM boot eFUSE descriptions

Fuse	Config	Definition	GPIO	Shipped value	Settings
EEPROM_RECOVE RITY_EN(0x4D0[24])	OEM	EEPROM recovery enable	No	0	0 - 3B (24-bit) 1 - 2B (16-bit)
SPI_ADDRESSING(0 x6D0[27])	OEM	SPI addressing (SPI only)	No	0	0 - 2 B (16-bit) 1 - 3 B (24-bit)
LPSPI_PORT_SEL(0 x6D0[26:25])	OEM	Port select	No	00	00 - LPSPI1 01 - LPSPI2 10 - LPSPI3 11 - LPSPI4
LPSPI_SPEED (0x6D0[29:28])	OEM	LPSPI Speed select	No	00	00 - 20 MHz 01 - 10 MHz 10 - 5 MHz 11 - 2 MHz

The LPSPI1/2/3/4 block can be used as a boot device using the LPSPI interface for the serial ROM boot. The SPI interface is configured to operate at speed specified by LPSPI_SPED_SEL fuse field.

The boot ROM copies 4 KB of data from the serial ROM device to the internal RAM. After checking the Image Vector Table header value (0xD1) from the program image, the ROM code performs a DCD check. After a successful DCD extraction, the ROM code extracts the destination pointer and length of image from the Boot Data Structure to be copied to the RAM device from where the code execution occurs.

NOTE

The Initial 4 KB of program image must contain the IVT, DCD, and the Boot Data Structures.

3.7 Program image

This section describes the data structures that are required to be included in the user's program image. The program image consists of:

- Image vector table—a list of pointers located at a fixed address that the ROM examines to determine where the other components of the program image are located.
- Boot data—a table that indicates the program image location, program image size in bytes, and the plugin flag.
- Device configuration data—IC configuration data.
- User code and data.

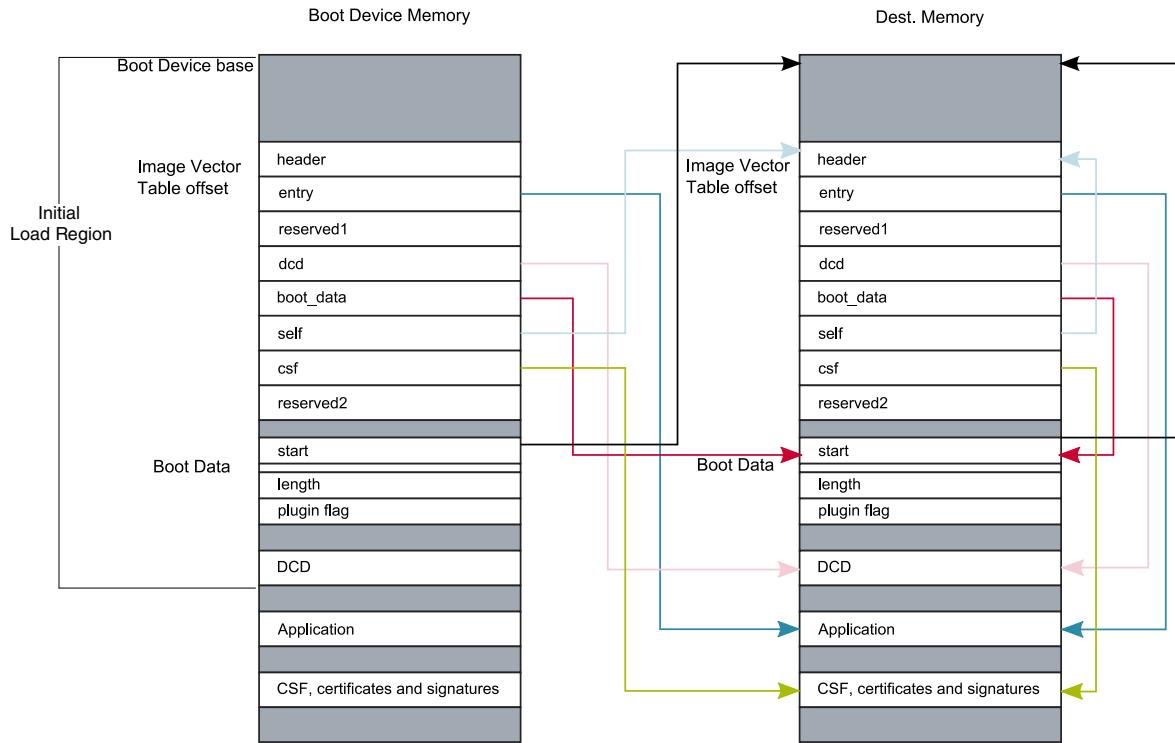
3.7.1 Image Vector Table and Boot Data

The Image Vector Table (IVT) is the data structure that the ROM reads from the boot device supplying the program image containing the required data components to perform a successful boot.

The IVT includes the program image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process. The ROM locates the IVT at a fixed address that is determined by the boot device connected to the Chip. The IVT offset from the base address and initial load region size for each boot device type is defined in the table below. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 3-42. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset	Initial Load Region Size
FlexSPI NOR/SEMC NOR	4 Kbyte = 0x1000 bytes	Entire Image Size
SD/MMC/eSD/eMMC/SDXC	1 Kbyte = 0x400 bytes	4 Kbyte
SPI NOR/EEPROM/SEMC NAND/ FlexSPI NAND	1 Kbyte = 0x400 bytes	4 Kbyte

**Figure 3-15. Image Vector Table**

3.7.1.1 Image vector table structure

The IVT has the following format where each entry is a 32-bit word:

Table 3-43. IVT format

header
entry: Absolute address of the first instruction to execute from the image
reserved1: Reserved and should be zero
dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on the DCD.
boot data: Absolute address of the boot data
self: Absolute address of the IVT. Used internally by the ROM.
csf: Absolute address of the Command Sequence File (CSF) used by the HAB library. See High-Assurance Boot (HAB) for details on the secure boot using HAB. This field must be set to NULL when not performing a secure boot
reserved2: Reserved and should be zero

Figure 3-16 shows the IVT header format:

Tag	Length	Version
-----	--------	---------

Figure 3-16. IVT header format

where:

Tag: A single byte field set to 0xD1

Length: a two byte field in big endian format containing the overall length of the IVT, in bytes, including the header. (the length is fixed and must have a value of 32 bytes)

Version: A single byte field set to 0x40 or 0x41

3.7.1.2 Boot data structure

The boot data must follow the format defined in the table found here, each entry is a 32-bit word.

Table 3-44. Boot data format

start	Absolute address of the image
length	Size of the program image
plugin	Plugin flag (see Plugin image)

3.7.2 Device Configuration Data (DCD)

Upon reset, the chip uses the default register values for all peripherals in the system. However, these settings typically are not ideal for achieving the optimal system performance and there are even some peripherals that must be configured before they can be used.

The DCD is a configuration information contained in the program image (external to the ROM) that the ROM interprets to configure various peripherals on the chip.

For example, some components (such as SDRAM) require some sequence of register programming as a part of the configuration before it is ready to be used. The DCD feature can be used to program the SEMC register to the optimal settings.

The ROM determines the location of the DCD table based on the information located in the Image Vector Table (IVT). See [Image Vector Table and Boot Data](#) for more details. The DCD table shown below is a big-endian byte array of the allowable DCD commands. The maximum size of the DCD is limited to 1768 B.

Header
[CMD]
[CMD]
...

Figure 3-17. DCD data format

The DCD header is 4 B with the following format:

Tag	Length	Version
-----	--------	---------

Figure 3-18. DCD header format

where:

Tag: A single-byte field set to 0xD2
 Length: a two-byte field in the big-endian format containing the overall length of the DCD (in bytes) including the header
 Version: A single-byte field set to 0x41

3.7.2.1 Write data command

The write data command is used to write a list of given 1-, 2- or 4-byte values (or bitmasks) to a corresponding list of target addresses.

The format of the write data command (in a big-endian byte array) is shown in this table:

Table 3-45. Write data command format

Tag	Length	Parameter
	Address	
	Value/Mask	
	[Address]	
	[Value/Mask]	
	...	
	[Address]	
	[Value/Mask]	

where:

Tag: a single-byte field set to 0xCC

Length: a two-byte field in a big-endian format, containing the length of the Write Data Command (in bytes) including the header

Address: the target address to which the data must be written

Value/Mask: the data value (or bitmask) to be written to the preceding address

The parameter field is a single byte divided into the bitfields, as follows:

Table 3-46. Write data command parameter field

7	6	5	4	3	2	1	0
flags						bytes	

where

bytes: the width of the target locations in bytes (either 1, 2, or 4)

flags: control flags for the command behavior

Data Mask = bit 3: if set, only specific bits may be overwritten at the target address (otherwise all bits may be overwritten)

Data Set = bit 4: if set, the bits at the target address are overwritten with this flag (otherwise it is ignored)

One or more target address and value/bitmask pairs can be specified. The same bytes' and flags' parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 3-47. Interpretation of write data command flags

"Mask"	"Set"	Action	Interpretation
0	0	*address = val_msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address = val_msk	Set bitmask

NOTE

If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.

If any of the values are larger or any of the bitmasks are wider than permitted by the data width indicated in the parameter field, none of the values are written.

If any of the target addresses do not lie within the allowed region, none of the values are written. The list of allowable blocks and target addresses for the chip are provided below.

3.7.2.2 Check data command

The check data command is used to test for a given 1-, 2-, or 4-byte bitmasks from a source address.

The check data command is a big-endian byte array with the format shown in this table:

Table 3-48. Check data command format

Tag	Length	Parameter
	Address	
	Mask	
	[Count]	

where:

Tag: a single-byte field set to 0xCF
 Length: a two-byte field in the big-endian format containing the length of the check data command (in bytes) including the header
 Address: the source address to test
 Mask: the bit mask to test
 Count: an optional poll count; If the count is not specified, this command polls indefinitely until the exit condition is met. If count = 0, this command behaves as for the NOP.

The parameter field is a single byte divided into bitfields, as follows:

Table 3-49. Check data command parameter field

7	6	5	4	3	2	1	0
flags						bytes	

where

bytes: the width of target locations in bytes (either 1, 2, or 4)
 flags: control flags for the command behavior
 Data Mask = bit 3: if set, only the specific bits may be overwritten at a target address (otherwise all bits may be overwritten)
 Data Set = bit 4: if set, the bits at the target address are overwritten with this flag (otherwise it is ignored)

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 3-50. Interpretation of check data command flags

"Mask"	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set

Table continues on the next page...

Table 3-50. Interpretation of check data command flags (continued)

"Mask"	"Set"	Action	Interpretation
1	0	(*address & mask)!= mask	Any bit clear
1	1	(*address & mask)!= 0	Any bit set

NOTE

If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.

If the bitmask is wider than permitted by the data width indicated in the parameter field, the value is not read.

3.7.2.3 NOP command

This command has no effect.

The format of the NOP command is a big-endian four-byte array, as shown in this table:

Table 3-51. NOP command format

Tag	Length	Undefined
-----	--------	-----------

where:

Tag: a single-byte field set to 0xC0
 Length: a two-byte field in big endian containing the length of the NOP command in bytes (fixed to a value of 4)
 Undefined: this byte is ignored and can be set to any value.

3.7.2.4 Unlock command

The unlock command is used to prevent specific engine features from being locked when exiting the ROM.

The format of the unlock command (in a big-endian byte array) is shown in this table:

Table 3-52. Unlock command format

Tag	Length	Eng
	Value	
	Value	
	...	

Table continues on the next page...

Table 3-52. Unlock command format (continued)

Value

where:

NOTE

This command may not be used in the DCD structure if the SEC_CONFIG is configured as closed.

3.8 Plugin image

The ROM supports a limited number of boot devices. When using other devices as a boot source (for example, Ethernet, CDROM, or USB), the supported boot device must be used (typically serial ROM) as a firmware to provide the missing boot drivers. Additionally, the plugin can be customized to support boot drivers, which is more flexible when performing the device initialization, such as condition judging, delay assertion, or to apply custom settings to the boot device and memory system.

In addition to the standard images, the chip also supports plugin images. The plugin images return the execution to the ROM whereas the standard image does not.

The boot ROM detects the image type using the plugin flag of the boot data structure (see [Boot data structure](#)). If the plugin flag is 1, then the ROM uses the image as a plugin function. The function must initialize the boot device and copy the program image to the final location. At the end, the plugin function must return with the program image parameters. (See [High-level boot sequence](#) for details about the boot flow).

The boot ROM authenticates the plugin image before running the plugin function and then authenticates the program image.

The plugin function must follow the API described below:

```
typedef BOOLEAN (*plugin_download_f)(void **start, size_t *bytes, UINT32
*iwt_offset);
```

ARGUMENTS PASSED:

- start - the image load address on exit.
- bytes - the image size on exit.
- iwt_offset - the offset (in bytes) of the IVT from the image start address on exit.

RETURN VALUE:

- 1 - success
- 0 - failure

3.9 Serial Downloader

The Serial Downloader provides a means to download a program image to the chip over the USB and UART serial connection.

In this mode, the ROM programs the WDOG1 for a time-out specified by the fuse WDOG Time-out Select (See the Fusemap chapter for details) if the WDOG_ENABLE eFuse is 1 and continuously polls for the USB and UART connection. If no activity is found on the USB OTG1 and UART and the watchdog timer expires, the ARM core is reset.

NOTE

After the downloaded image is loaded, it is responsible for managing the watchdog resets properly.

This figure shows the USB and UART boot flow:

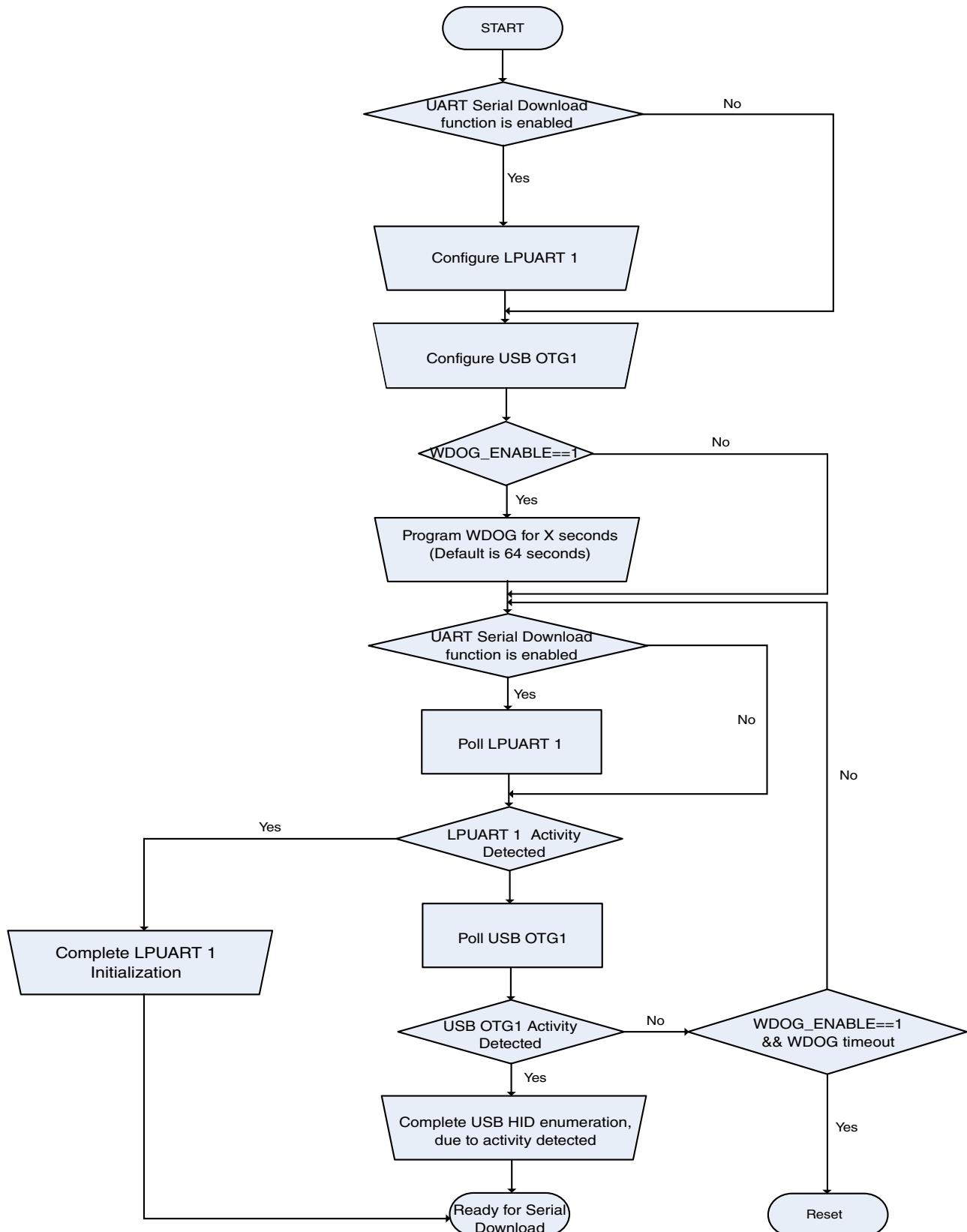


Figure 3-19. Serial Downloader boot flow

3.9.1 USB

The USB support is composed of the USBOH3(USB OTG1 core controller, compliant with the USB 2.0 specification) and the USBPHY (HS USB transceiver).

The ROM supports the USB OTG port for boot purposes. The other USB ports on the chip are not supported for boot purposes.

The USB Driver is implemented as a USB HID class. A collection of four HID reports are used to implement the SDP protocol for data transfers, as described in [Table 3-53](#).

Table 3-53. USB HID reports

Report ID (first byte)	Transfer endpoint	Direction	Length	Description
1	control OUT	Host to device	17 B	SDP command from the host to the device.
2	control OUT	Host to device	Up to 1025 B	Data associated with the report 1 SDP command.
3	interrupt	Device to host	5 B	HAB security configuration. The device sends 0x12343412 in the closed mode and 0x56787856 in the open mode.
4	interrupt	Device to host	Up to 65 B	Data in response to the SDP command in report 1.

3.9.1.1 USB configuration details

The USB OTG function device driver supports a high speed (HS for UTMI) non-stream mode with a maximal packet size of 512 B and a low-level USB OTG function.

The VID/PID and strings for the USB device driver are listed in the following table.

Table 3-54. VID/PID and strings for USB device driver

Descriptor	Value
VID	0x15A2 (NXP vendor ID)
PID ¹	
String Descriptor1 (manufacturer)	NXP Semiconductors
String Descriptor2 (product)	S Blank

Table continues on the next page...

Table 3-54. VID/PID and strings for USB device driver (continued)

Descriptor	Value
String Descriptor4	NXP Flash
String Descriptor5	NXP Flash

1. Allocation based on the BPN (Before Part Number)

3.9.1.2 IOMUX configuration for USB

The interface signals of the UTMI PHY are not configured in the IOMUX. The UTMI PHY interface uses the dedicated contacts on the IC. See the chip data sheet for details.

3.9.2 Serial Download Protocol (SDP)

The 16-byte SDP command from the host to device is sent using the HID report 1. This table describes the 16-byte SDP command data structure:

Table 3-55. 16-byte SDP command data structure

BYTE offset	Size	Name	Description
0	2	COMMAND TYPE	These commands are supported for the ROM: <ul style="list-style-type: none"> • 0x0101 READ_REGISTER • 0x0202 WRITE_REGISTER • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0xA0A DCD_WRITE • 0xB0B JUMP_ADDRESS
2	4	ADDRESS	Only relevant for these commands: READ_REGISTER, WRITE_REGISTER, WRITE_FILE, DCD_WRITE, and JUMP_ADDRESS. For the READ_REGISTER and WRITE_REGISTER commands, this field is the address to a register. For the WRITE_FILE and JUMP_ADDRESS commands, this field is an address to the internal or external memory address.
6	1	FORMAT	Format of access, 0x8 for an 8-bit access, 0x10 for a 16-bit access, and 0x20 for a 32-bit access. Only relevant for the READ_REGISTER and WRITE_REGISTER commands.

Table continues on the next page...

Table 3-55. 16-byte SDP command data structure (continued)

BYTE offset	Size	Name	Description
7	4	DATA COUNT	Size of the data to read or write. Only relevant for the WRITE_FILE, READ_REGISTER, WRITE_REGISTER, and DCD_WRITE commands. For the WRITE_FILE and DCD_WRITE commands, the DATA COUNT is in the byte units.
11	4	DATA	The value to write. Only relevant for the WRITE_REGISTER command.
15	1	RESERVED	Reserved

3.9.2.1 SDP commands

The SDP commands are described in the following sections.

3.9.2.1.1 READ_REGISTER

The transaction for the READ_REGISTER command consists of these reports: Report1 for the command, Report3 for the security configuration, and Report4 for the response or the register value.

The register to read is specified in the ADDRESS field of the SDP command. The first device sends Report3 with the security configuration followed by the Report4 with the bytes read at a given address. If the count is greater than 64, multiple reports with the report id 4 are sent until the entire data requested by the host is sent. The STATUS is either 0x12343412 for the closed parts and 0x56787856 for the open or field return parts.

Report1, Command, Host to Device:

1	Valid values for the READ_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT
---	---

ID 16-byte SDP command

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration
---	--

ID 4 bytes status

Report4, Response, Device to Host: first response report

4	Register value
---	----------------

ID 4 bytes of data containing the register value. If the number of bytes requested is less than 4, the remaining bytes must be ignored by the host.

Multiple reports of the report id 4 are sent until the entire requested data is sent.

Report4, Response, Device to Host: last response report

4	Register value
---	----------------

ID 64 bytes of data containing the register value. If the number of bytes requested is less than 64, the remaining bytes must be ignored by the host.

3.9.2.1.2 WRITE_REGISTER

The transaction for the WRITE_REGISTER command consists of these reports: Report1 for the command, Report3 for the security configuration and Report4 for the write status.

The host sends Report1 with the WRITE_REGISTER command. The register to write is specified in the ADDRESS field of the SDP command of Report1, with the FORMAT field set to the data type (number of bits to write, either 8, 16, or 32) and the value to write in the DATA field of the SDP command. The device writes the DATA to the register address and returns the WRITE_COMPLETE code using Report4 and the security configuration using Report3 to complete the transaction.

Report1, Command, Host to Device:

1	Valid values for WRITE_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT and DATA
---	---

ID 16-byte SDP command

Report3, Response, Device to Host:

3	4 bytes indicating the security configuration
---	---

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status
---	------------------------------------

ID 64 bytes data with the first 4 bytes to indicate that the write is completed with code 0x128A8A12. On failure, the device reports the HAB error status.

3.9.2.1.3 WRITE_FILE

The transaction for the WRITE_FILE command consists of these reports: Report1 for the command phase, Report2 for the data phase, Report3 for the HAB mode, and Report4 to indicate that the data are received in full.

The size of each Report2 is limited to 1024 bytes (limitation of the USB HID protocol). Hence, multiple Report2 packets are sent by the host in the data phase until the entire data is transferred to the device. When the entire data (DATA_COUNT bytes) is received, the device sends Report3 with the HAB mode and Report4 with 0x88888888, indicating that the file download completed.

Report1, Host to Device:

1	Valid values for WRITE_FILE COMMAND, ADDRESS, DATA_COUNT
---	--

ID 16-byte SDP command

=====Optional Begin=====

Host sends the ERROR_STATUS command to query if the HAB rejected the address

===== Optional End=====

Report2, Host to Device:

2	File data
---	-----------

ID Max 1024 bytes data per report

Report2, Host to Device:

2	File data
---	-----------

ID Max 1024 bytes data per report

Report3, Device to Host:

3	4 bytes indicating security configuration
---	---

ID 4 bytes status

Report4, Response, Device to Host:

4	COMPLETE (0x88888888) status
---	------------------------------

ID 64 bytes data with the first four bytes to indicate that the file download completed with code 0x88888888. On failure, the device reports the HAB error status.

3.9.2.1.4 ERROR_STATUS

The transaction for the SDP command ERROR_STATUS consists of three reports.

Report1 is used by the host to send the command; the device sends global error status in four bytes of Report4 after returning the security configuration in Report3. When the device receives the ERROR_STATUS command, it returns the global error status that is updated for each command. This command is useful to find out whether the last command resulted in a device error or succeeded.

Report1, Command, Host to Device:

1	ERROR_STATUS COMMAND
---	----------------------

ID 16-byte SDP Command

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration
---	--

ID 4 bytes status

Report4, Response, Device to Host:

4	Four bytes Error status
---	-------------------------

ID first 4 bytes status in 64 bytes Report4

3.9.2.1.5 DCD_WRITE

The SDP command DCD_WRITE is used by the host to send multiple register writes in one shot. This command is provided to speed up the process of programming the register writes (such as to configure an external RAM device).

The command goes with Report1 from the host with COMMAND TYPE set to DCD_WRITE, ADDRESS which is used as a temporary location of the DCD data, and DATA_COUNT to the number of bytes sent in the data out phase. In the data phase, the

host sends the data for a number of registers using Report2. The device completes the transaction with Report3 indicating the security configuration and Report4 with the WRITE_COMPLETE code 0x128A8A12.

Report1, Command, Host to Device:

1	DCD_WRITE COMMAND, ADDRESS, DATA_COUNT
---	--

ID 16-byte SDP Command

Report2, Data, Host to Device:

2	DCD binary data
---	-----------------

ID Max 1024 bytes per report

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration
---	--

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status
---	------------------------------------

ID 64 bytes report with the first four bytes to indicate that the write completed with the code 0x128A8A12. On failure, the device reports the HAB error status.

See [Device Configuration Data \(DCD\)](#) for the DCD format description.

3.9.2.1.6 JUMP_ADDRESS

The SDP command JUMP_ADDRESS is the last command that the host can send to the device. After this command, the device jumps to the address specified in the ADDRESS field of the SDP command and starts to execute.

This command usually follows after the WRITE_FILE command. The command is sent by the host in the command-phase of the transaction using Report1. There is no data phase for this command, but the device sends the status Report3 to complete the transaction. If the authentication fails, it also sends Report4 with the HAB error status.

Report1, Command, Host to Device:

Recovery devices

1	JUMP_ADDRESS COMMAND, ADDRESS
---	-------------------------------

ID 16-byte SDP Command

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration
---	--

ID 4 bytes status

This report is sent by the device only in case of an error jumping to the given address, or if the device reports error in Report4, Response, Device to Host:

4	Four bytes HAB error status
---	-----------------------------

ID 4 bytes status, 64 bytes report length

3.10 Recovery devices

The chip supports recovery devices. If the primary boot device fails, the boot ROM tries to boot from the recovery device using one of the LPSPI ports.

To enable the recovery device, the EEPROM_RECOVERY_EN fuse must be set.

Additionally, the serial EEPROM fuses must be set as described in [Serial NOR/EEPROM through LPSPI](#).

3.11 SD/MMC manufacture mode

When the internal boot and recover boot (if enabled) failed, the boot goes to the SD/MMC manufacture mode before the serial download mode. In the manufacture mode, one bit bus width is used despite of the fuse setting.

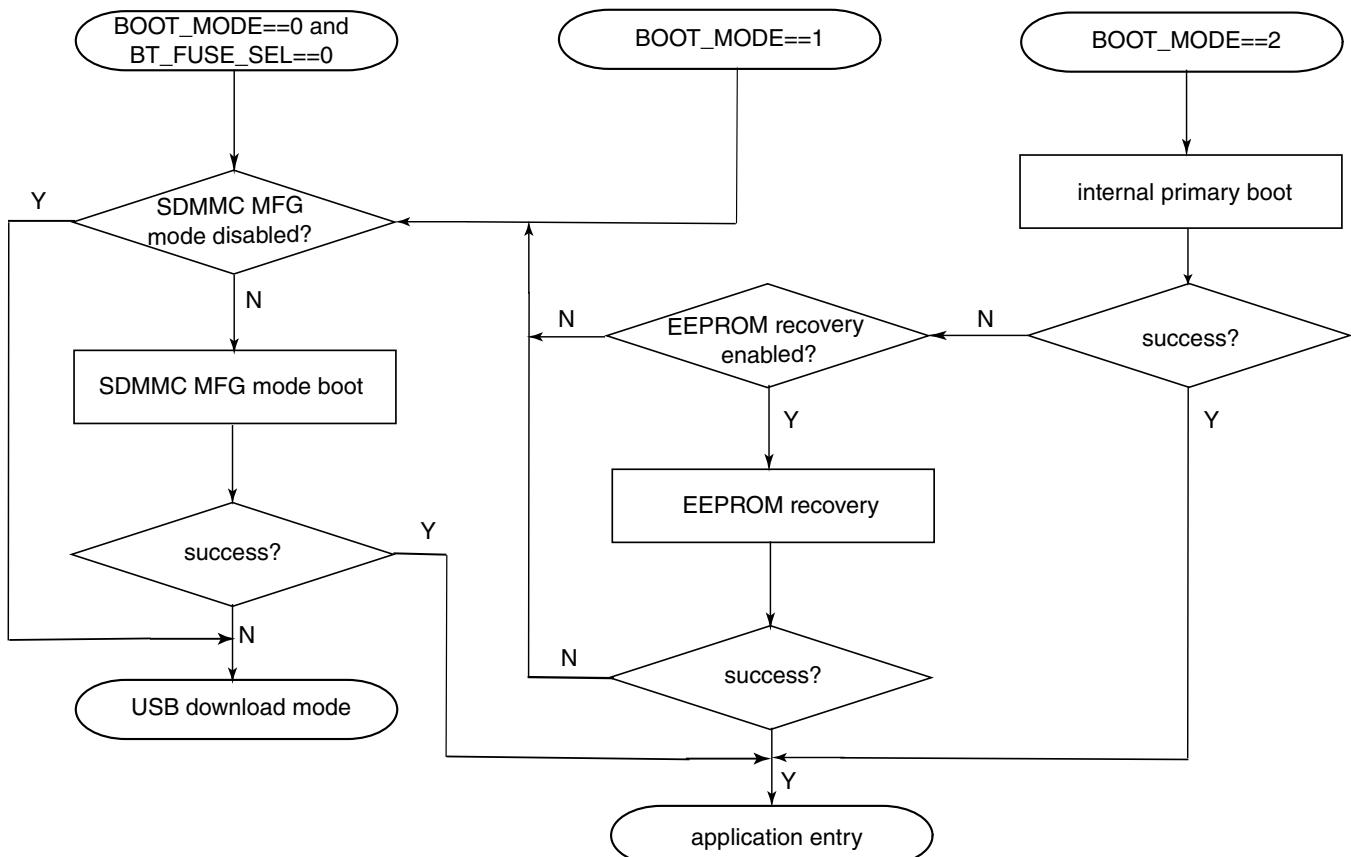


Figure 3-20. SD/MMC manufacture boot flow

3.12 High-Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying the areas of code or data in the programmable memory to make it behave in an incorrect manner. The HAB also prevents the attempts to gain access to features which must not be available.

The integration of the HAB feature with the ROM code ensures that the chip does not enter an operational state if the existing hardware security blocks detected a condition that may be a security threat or if the areas of memory deemed to be important were modified. The HAB uses the RSA digital signatures to enforce these policies.

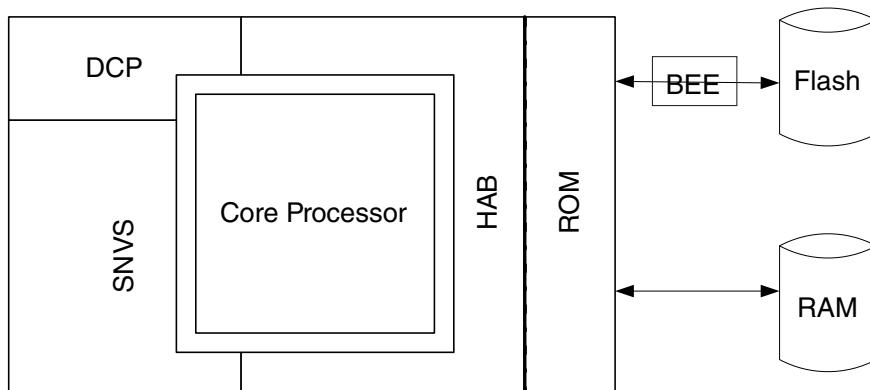


Figure 3-21. Secure boot components

The figure above illustrates the components used during a secure boot using HAB. The HAB interfaces with the SNVS to make sure that the system security state is as expected. The HAB also uses the hardware block to accelerate the SHA-256 message digest operations performed during the signature verifications. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator can't be used. The RSA key sizes supported are 1024, 2048, 3072 and 4096 bits. The RSA signature verification operations are performed by a software implementation contained in the HAB library. The main features supported by the HAB are:

- X.509 public key certificate support
- CMS signature format support

NOTE

NXP provides the reference Code Signing Tool (CST) for key generation and code signing for use with the HAB library. The CST can be found by searching for "IMX_CST_TOOL" at <http://www.nxp.com>.

NOTE

For further details on using the secure boot feature using HAB, contact your local NXP representative.

3.12.1 HAB API vector table addresses

For devices that perform a secure boot, the HAB library may be called by the boot stages that execute after the ROM code.

The API vector table for this device is at address 0x0020_0300.

NOTE

For additional information on the secure boot including the HAB API, contact your local NXP representative.

Chapter 4

Fusemap

4.1 Boot Fusemap

This section describes the various modes and the selection of the required boot devices.

A separate map is given for each and every boot device. The device select is specified by the BOOT_CFG1[7:4] fuses.

Table 4-1. Boot device select

Boot Device	BOOT_CFG1[7]	BOOT_CFG1[6]	BOOT_CFG1[5]	BOOT_CFG1[4]
FlexSPI1 (Serial NOR)	0	0	0	0
SD	0	1	x	x
MMC/eMMC	1	0	x	x
SEMC (NAND)	0	0	1	x
SEMC (NOR)	0	0	0	1
FlexSPI1 (Serial NAND)	1	1	x	x

NOTE

The fuses marked as “Reserved” are reserved for NXP internal (and future) use only. Customers **must not** attempt to burn these, because the IC behavior may be unpredictable. The reserved fuses can be read as either '0' or '1'.

Table 4-2. FlexSPI1 (Serial NOR) boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	0	0	HOLD TIME 00 - 500us 01 - 1 ms 10 - 3ms 11 - 10ms	EncryptedXI P 0 - Disabled 1 - Enabled	Reserved	

Table continues on the next page...

Table 4-2. FlexSPI1 (Serial NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Infinite loop: (debug use only) 0 - Disable 1 - Enable	FLASH TYPE: 000 - Device supports 3B read by default 001 - Device supports 4B read by default 010 - HyperFlash 1V8 011 - HyperFlash 3V3 100 - MXIC Octal DDR 101 - Micron Octal DDR 111 - QSPI device supports 3B read by default (on secondary pinmux option)		
0x450[23:16] (BOOT_CFG3)					Reserved			
0x450[31:24] (BOOT_CFG4)					Reserved			
0x460[7:0]	Reserved	FORCE_C OLD_BOO T(SBMR)	BT_FUSE_ SEL	DIR_BT_DI S	Boot Frequencies (ARM/BSU) 0 - 396/132 MHz 1 - 528/132 MHz	SEC_CONF IG[1]	Reserved	
0x460[15:8]	BEE_KEY1_SEL	BEE_KEY0_SEL			Reserved (SDR config)			
0x460[23:16]	JTAG_SMODE[1:0]	WDOG_EN ABLE '0' - Disabled '1' - Enabled	SJC_DISAB LE	DAP_SJC_ SWD_SEL	SDP_READ _DISABLE	SDP_DISA BLE	FORCE_IN TERNAL_B OOT	
0x460[31:24]	SD_PWR_CYCLE_SELE CTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms	PWR_STAB LE_CYCLE _SELECTIO N: '0' - 5ms '1' - 2.5ms	Reserved	JTAG_HEO	KTE	NAND_ECC _DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENAB LE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc	
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTIO N 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufactur e mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTI NGS value)

Table continues on the next page...

Table 4-2. FlexSPI1 (Serial NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
	1 - DLL Override Mode for SD/eMMC							
0x470[15:8]	SD2_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_ PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO MUX_SION _BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	USDHC_C MD_OE_PR E_EN (SD/MMC debug)	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTI NGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 4-3. SD boot fusemap

Address	7	6	5	4	3	2	1	0	
0x450[7:0] (BOOT_CFG1)	0	1	SD/SDXC Speed: 00 - Normal/SDR12 01 - High/SDR25 10 - SDR50 11 - SDR104			SD Power Cycle Enable: '0' - No power cycle '1' - Enabled via USDHC_RS T pad	SD Loop back Clock Source Select: (for SDR50 and SDR104 only) '0' - through SD pad '1' - direct	Port Select: 0 - eSDHC1 1 - eSDHC2	Fast Boot: 0 - Regular 1 - Fast Boot
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Infinite loop: (debug use only) 0 - Disable 1 - Enable	Reserved	Bus Width: 0 - 1-bit 1 - 4-bit	SD1 Voltage Selection: 0 - 3.3V 1 - 1.8V	
0x450[23:16]	Reserved								

Table continues on the next page...

Table 4-3. SD boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
(BOOT_CFG3)								
0x450[31:24]					Reserved			
(BOOT_CFG4)								
0x460[7:0]	Reserved	FORCE_C OLD_BOO T(SBMR)	BT_FUSE_ SEL	DIR_BT_DI S	Boot Frequencies (ARM/BSU) 0 - 396/132 MHz 1 - 528/132 MHz	SEC_CONF IG[1]	Reserved	
0x460[15:8]	BEE_KEY1_SEL	BEE_KEY0_SEL			Reserved (SDR config)			
0x460[23:16]	JTAG_SMODE[1:0]	WDOG_EN ABLE '0' - Disabled '1' - Enabled	SJC_DISAB LE	DAP_SJC_ SWD_SEL	SDP_READ _DISABLE	SDP_DISA BLE	FORCE_IN TERNAL_B OOT	
0x460[31:24]	SD_PWR_CYCLE_SELE CTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms	PWR_STAB LE_CYCLE _SELECTIO N: '0' - 5ms '1' - 2.5ms	Reserved	JTAG_HEO	KTE	NAND_ECC _DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENAB LE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc	
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTIO N	UART Serial Download Disable: 0 - 3.3V 1 - 1.8V	Disable SDMMC Manufactur e mode: '0' - Not Disable '1' - Disable	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTI NGS value)
0x470[15:8]	SD2_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_ PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO MUX_SION _BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	USDHC_C MD_OE_PR E_EN (SD/MMC debug)	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4	Reserved		Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ...			

Table continues on the next page...

Table 4-3. SD boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
		'11' - Div by 8			11111 - gpio1.IO31			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTI NGS value)				Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			

MMC_DLL_DLY[6:0]:
Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value.

DELAY_CELL_NUM (FlexSPI NOR):
"000" - DLL Override feature is disabled
"001-111" - DLL Override feature is enabled,
See OVRDVAL in FLEXSPI chapter in RM for more details

Table 4-4. MMC/eMMC boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	1	0	SD/MMC Speed: 0 - Normal 1 - High	Fast Boot Acknowledg e Enable: 0 - Boot Ack Disabled 1 - Boot Ack Enabled	SD Power Cycle Enable: '0' - No power cycle '1' - Enabled via USDHC_RS T pad	SD Loop back Clock Source Select: (for SDR50 and SDR104 only) '0' - through SD pad '1' - direct	Port Select: 0 - eSDHC1 1 - eSDHC2	Fast Boot: 0 - Regular 1 - Fast Boot
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Infinite loop: (debug use only) 0 - Disable 1 - Enable	Bus Width: 01 - 4-bit 01 - 8-bit 10 - 4-bit DDR(MMC 4.4) 11 - 8-bit DDR(MMC 4.4)	SD1 Voltage Selection: 0 - 3.3V 1 - 1.8V	
0x450[23:16] (BOOT_CFG3)					Reserved			
0x450[31:24] (BOOT_CFG4)					Reserved			
0x460[7:0]	Reserved	FORCE_C OLD_BOO T(SBMR)	BT_FUSE_SEL	DIR_BT_DI S	Boot Frequencies (ARM/BSU) 0 - 396/132 MHz 1 - 528/132 MHz	SEC_CONF IG[1]	Reserved	
0x460[15:8]	BEE_KEY1_SEL	BEE_KEY0_SEL			Reserved (SDR config)			

Table continues on the next page...

Table 4-4. MMC/eMMC boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
0x460[23:16]	JTAG_SMODE[1:0]	WDOG_EN ABLE '0' - Disabled '1' - Enabled	SJC_DISAB LE	DAP_SJC_ SWD_SEL	SDP_READ_ _DISABLE	SDP_DISA BLE	FORCE_IN TERNAL_B OOT	
0x460[31:24]	SD_PWR_CYCLE_SELE CTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms	PWR_STAB LE_CYCLE _SELECTIO N: '0' - 5ms '1' - 2.5ms	Reserved	JTAG_HEO	KTE	NAND_ECC _DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENAB LE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc	
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTIO N 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufactur e mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTI NGS value)
0x470[15:8]	SD2_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_ PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO MUX_SION _BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	USDHC_C MD_OE_PR E_EN (SD/MMC debug)	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8	Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 <p>Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default</p>				
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTI NGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 4-5. SEMC (NAND) boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	1	BOOT_SEARCH_STRIDE Search stride for FCB and DBBT Search strides in terms of page 0000 - 64 Other value - $2^{(BOOT_SEARCH_STRIDE)}$				BOOT_SEA RCH_COU NT 0 - 1 1 - 2
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Infinite loop: (debug use only) 0 - Disable 1 - Enable	DQS disable: 0 - disabled 1 - enabled	ECC_ALG_ SEL Disable: 0 - SW ECC selected 1 - Device ECC selected	ONFI compliant: 0 - Yes, ONFI 1 - No, spec
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved							
0x460[7:0]	Reserved	FORCE_C OLD_BOO T(SBMR)	BT_FUSE_ SEL	DIR_BT_DI S	Boot Frequencies (ARM/BSU) 0 - 396/132 MHz 1 - 528/132 MHz	SEC_CONF IG[1]	Reserved	
0x460[15:8]	BEE_KEY1_SEL		BEE_KEY0_SEL		Reserved (SDR config)			
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_EN ABLE '0' - Disabled '1' - Enabled	SJC_DISAB LE	DAP_SJC_ SWD_SEL	SDP_READ _DISABLE	SDP_DISA BLE	FORCE_IN TERNAL_B OOT
0x460[31:24]	SD_PWR_CYCLE_SELE CTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STAB LE_CYCLE _SELECTIO N: '0' - 5ms '1' - 2.5ms	Reserved	JTAG_HEO	KTE	NAND_ECC _DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENAB LE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low	SD2 VOLTAGE SELECTIO N 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable	Disable SDMMC Manufactur e mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTI NGS value)

Table continues on the next page...

Table 4-5. SEMC (NAND) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
	1 - DLL Override Mode for SD/eMMC	1 - Reset active high		'1' - Disable				
0x470[15:8]	SD2_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA_D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO_MUX_SION_BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC_IOMUX_SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	USDHC_C MD_OE_PR E_EN (SD/MMC debug)	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8	Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default				
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTINGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 4-6. SEMC (NOR) boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	0	1	Reserved	Clock Freq: 000 - 33MHz 001 - 66MHz 010 - 108MHz 011 - 133MHz 1xx - 166MHz		
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Infinite loop: (debug use only) 0 - Disable 1 - Enable	DQS Disable: 0 - disabled 1 - enabled	Data Port Size: 0 - 16 bit 1 - 8 bit	AC Timing 0 - Default 1 - Fuse (0x6E0)
0x450[23:16]	Reserved							

Table continues on the next page...

Table 4-6. SEMC (NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
(BOOT_CFG3)								
0x450[31:24]					Reserved			
(BOOT_CFG4)								
0x460[7:0]	Reserved	FORCE_C OLD_BOO T(SBMR)	BT_FUSE_ SEL	DIR_BT_DI S	Boot Frequencies (ARM/BSU) 0 - 396/132 MHz 1 - 528/132 MHz	SEC_CONF IG[1]	Reserved	
0x460[15:8]	BEE_KEY1_SEL	BEE_KEY0_SEL			Reserved (SDR config)			
0x460[23:16]	JTAG_SMODE[1:0]	WDOG_EN ABLE '0' - Disabled '1' - Enabled	SJC_DISAB LE	DAP_SJC_ SWD_SEL	SDP_READ _DISABLE	SDP_DISA BLE	FORCE_IN TERNAL_B OOT	
0x460[31:24]	SD_PWR_CYCLE_SELE CTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms	PWR_STAB LE_CYCLE _SELECTIO N: '0' - 5ms '1' - 2.5ms	Reserved	JTAG_HEO	KTE	NAND_ECC _DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENAB LE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc	
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTIO N	UART Serial Download Disable: 0 - 3.3V 1 - 1.8V	Disable SDMMC Manufactur e mode: '0' - Not Disable '1' - Disable	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTI NGS value)
0x470[15:8]	SD2_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_ PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO MUX_SION _BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	USDHC_C MD_OE_PR E_EN (SD/MMC debug)	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4	Reserved		Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ...			

Table continues on the next page...

Table 4-6. SEMC (NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
		'11' - Div by 8			11111 - gpio1.IO31			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTI NGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details			Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			

Table 4-7. FlexSPI1 (Serial NAND) boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	1	1	SAFE FREQ: (Default safe communication frequency) 0 – High Speed (50MHz) 1 – Low Speed (30MHz)	COL_ADDR_ESS_WIDTH: H: 0 – 12bits 1 – 13bits	SPI NAND HOLD TIME: 00 - 0us 01 - 500us 10 - 1ms 11 - 3ms	BOOT_SEARCH_STRIDE: Search stride for FCB and DBBT (in terms of page) 0 - 64 1 - 128 2 - 256 3 - 32		
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Infinite loop: (debug use only) 0 - Disable 1 - Enable	CS_INTERVAL: CS de-asserted interval between two commands 0 – 100ns 1 – 200ns 2 – 400ns 3 – 50ns	BOOT_SEA RCH_COU NT: 0 - 1 1 - 2	
0x450[23:16] (BOOT_CFG3)					Reserved			
0x450[31:24] (BOOT_CFG4)					Reserved			
0x460[7:0]	Reserved	FORCE_C_OLD_BOOT(SBMR)	BT_FUSE_SEL	DIR_BT_DIS	Boot Frequencies (ARM/BSU) 0 - 396/132 MHz	SEC_CONFIG[1]	Reserved	

Table continues on the next page...

Table 4-7. FlexSPI1 (Serial NAND) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
						1 - 528/132 MHz		
0x460[15:8]	BEE_KEY1_SEL	BEE_KEY0_SEL	Reserved (SDR config)					
0x460[23:16]	JTAG_SMODE[1:0]	WDOG_EN '0' - Disabled '1' - Enabled	SJC_DISAB LE	DAP_SJC_ SWD_SEL	SDP_READ _DISABLE	SDP_DISA BLE	FORCE_IN TERNAL_B OOT	
0x460[31:24]	SD_PWR_CYCLE_SELE CTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms	PWR_STAB LE_CYCLE _SELECTIO N: '0' - 5ms '1' - 2.5ms	Reserved	JTAG_HEO	KTE	NAND_ECC _DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENAB LE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc	
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTIO N 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufactur e mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTI NGS value)
0x470[15:8]	SD2_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_ PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO MUX_SION _BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	USDHC_C MD_OE_PR E_EN (SD/MMC debug)	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTI NGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled						

Table 4-7. FlexSPI1 (Serial NAND) boot fusemap

Address	7	6	5	4	3	2	1	0
"001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details								

4.2 Lock Fusemap

Table 4-8. Lock fuses

Address	7	6	5	4	3	2	1	0
0x400[7:0]	Reserved	SJC_RESP_LOCK 0- Unlock '1' - WP,OP,RP	MEM_TRIM_LOCK 00- Unlock '1x' - OP 'x1' - WP	BOOT_CFG_LOCK 00- Unlock '1x' - OP 'x1' - WP	TESTER_LOCK 00 -Unlock '1x' - OP 'x1' - WP			
0x400[15:8]	OTPMK_M SB_LOCK 0- Unlock '1' - RP, WP, OP	SRK_LOCK 0- Unlock '1' - WP,OP,RP	GP2_LOCK 00- Unlock '1x' - OP 'x1' - WP	GP1_LOCK 00- Unlock '1x' - OP 'x1' - WP	MAC_ADDR_LOCK 00- Unlock '1x' - OP 'x1' - WP			
0x400[23:16]	SW_GP2_R LOCK '1' - RP of SW_GP2 fuses	MISC_CON F_LOCK '1' - WP + OP	SW_GP2_L OCK '1' - WP + OP of SW_GP2 fuses	Reserved	ANALOG_LOCK 00- Unlock '1x' - OP 'x1' - WP	Reserved	SW_GP1_L OCK 0- Unlock '1' - WP + OP of SW_GP1 fuses	
0x400[31:24]	FIELD_RET URN: '1' - Field Returned part '0' - functional part	Reserved	Reserved		GP3_LOCK '1x' - OP 'x1' - WP	Reserved		

4.3 Fusemap Descriptions Table

This section describes the chip fusemap descriptions.

Table 4-9. Fusemap Descriptions

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x400[1:0]	TESTER_LOCK	2	Perform lock on Tester programed fuses.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[3:2]	BOOT_CFG_LOC_K	2	Perform lock on BOOT related fuses.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[5:4]	MEM_TRIM_LOC_K	2	Trimming fuses. Burnt on the tester or by customer before the final product shipment.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[6]	SJC_RESP_LOC_K	1	SJC response lock	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can't be read, sensed, burned or overridden in the corresponded IIM register)	N/A
0x400[7]	Reserved	1	Reserved	Reserved	Reserved
0x400[9:8]	MAC_ADDR_LOC_K	2	Lock MAC_ADDR fuses.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register)	N/A

Table continues on the next page...

Fusemap Descriptions Table

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	
0x400[11:10]	GP1_LOCK	2	Lock for General Purpose fuse register #1 (GP1)	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[13:12]	GP2_LOCK	2	Lock for General Purpose fuse register #2 (GP2)	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[14]	SRK_LOCK	1	Locking SRK_HASH[255:0]	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can't be read, sensed, burned or overridden in the corresponded IIM register)	N/A
0x400[15]	OTPMK_MSB_LOCK	1	Locking OTPMK[255:128] - whole fusebank2. When blown, the fuses cannot be read, overridden nor written.	0 - Unlock 1 - Lock (OTPMKY[255:128] fuses, shadow register cannot be read, overridden nor written)	N/A
0x400[16]	SW_GP1_LOCK	1	Lock for SW_GP1 fuse.	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can't be sensed, burned or overridden in the corresponded IIM register)	N/A
0x400[17]	Reserved	1	Reserved	Reserved	Reserved

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x400[19:18]	ANALOG_LOCK	2	Lock for analog related fuse.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[20]	Reserved	1	Reserved	Reserved	Reserved
0x400[21]	SW_GP2_LOCK	1	WP and OP Lock for SW_GP2 fuse.	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can't be sensed, burned or overridden in the corresponded IIM register)	N/A
0x400[22]	MISC_CONF_LOCK	1			N/A
0x400[23]	SW_GP2_RLOCK	1	RP Lock for SW_GP2 fuse.	0 - Unlock (The controlled field can be read in the corresponded IIM register) 1 - Lock (The controlled field can't be read in the corresponded IIM register)	N/A
0x400[24]	Reserved	1	Reserved	Reserved	Reserved
0x400[25]	Reserved	1	Reserved	Reserved	Reserved
0x400[27:26]	SW_GP3_RLOCK	2	RP Lock for SW_GP3 fuse.	0 - Unlock (The controlled field can be read in the corresponded IIM register) 1 - Lock (The controlled field can't be read in the corresponded IIM register)	N/A
0x400[27:26]	GP3_LOCK	2	Lock for GP3 fuse	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[29:28]	Reserved	2	Reserved	Reserved	Reserved
0x400[30]	Reserved	1	Reserved	Reserved	Reserved
0x400[31]	FIELD_RETURN	1	Configure device for field return testing. Fuse burning is protected by CSF command, with proper parameter	0' - Device is in functional / secure mode. '1' - Device is open for 'field-return' testing.	N/A

Table continues on the next page...

Fusemap Descriptions Table

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
			passed. Write / OP protected by FIELD_RETURN_LOCK bit in control register.		
0x420[10:0], 0x410[31:0]	LOT_NO_ENC[42:0](SJC_CHALL/ UNIQUE_ID[42:0])	43	FSL-wide unique, encoded LOT ID STD II/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x420[15:11]	WAFER_NO[4:0] (SJC_CHALL[47:43] / UNIQUE_ID[47:43])	5	The wafer number of the wafer on which the device was fabricated/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x420[23:16]	DIE-Y-CORDINATE[7:0] (SJC_CHALL[55:48] / UNIQUE_ID[55:48])	8	The Y-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x420[31:24]	DIE-X-CORDINATE[7:0] (SJC_CHALL[63:56] / UNIQUE_ID[63:56])	8	The X-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x430[7:0]	Reserved	8	Reserved	Reserved	Reserved
0x430[15:8]	Reserved	8	Reserved	Reserved	Reserved
0x430[19:16]	SI_REV[3:0]	4	Silicon Revision number.		TESTER_LOCK
0x430[20]	Reserved	1	Reserved	Reserved	Reserved
0x430[21]	Reserved	1	Reserved	Reserved	Reserved
0x430[22]	Reserved	1	Reserved	Reserved	Reserved
0x430[23]	Reserved	1	Reserved	Reserved	Reserved
0x430[24]	Reserved	1	Reserved	Reserved	Reserved
0x430[25]	Reserved	1	Reserved	Reserved	Reserved
0x430[26]	Reserved	1	Reserved	Reserved	Reserved
0x430[27]	Reserved	1	Reserved	Reserved	Reserved
0x430[28]	Reserved	1	Reserved	Reserved	Reserved
0x430[29]	Reserved	1	Reserved	Reserved	Reserved
0x430[30]	Reserved	1	Reserved	Reserved	Reserved
0x430[31]	Reserved	1	Reserved	Reserved	Reserved
0x440[0]	Reserved	1	Reserved	Reserved	Reserved
0x440[1]	Reserved	1	Reserved	Reserved	Reserved
0x440[2]	Reserved	1	Reserved	Reserved	Reserved
0x440[3]	Reserved	1	Reserved	Reserved	Reserved

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x440[4]	Reserved	1	Reserved	Reserved	Reserved
0x440[5]	Reserved	1	Reserved	Reserved	Reserved
0x440[6]	Reserved	1	Reserved	Reserved	Reserved
0x440[7]	Reserved	1	Reserved	Reserved	Reserved
0x440[8]	Reserved	1	Reserved	Reserved	Reserved
0x440[9]	Reserved	1	Reserved	Reserved	Reserved
0x440[10]	Reserved	1	Reserved	Reserved	Reserved
0x440[11]	Reserved	1	Reserved	Reserved	Reserved
0x440[12]	Reserved	1	Reserved	Reserved	Reserved
0x440[13]	Reserved	1	Reserved	Reserved	Reserved
0x440[14]	Reserved	1	Reserved	Reserved	Reserved
0x440[15]	Reserved	1	Reserved	Reserved	Reserved
0x440[17:16]	SPEED_GRADIN G[1:0]	2	Burned by tester program, for indicating IC core speed	FRAL[0:1] MHz P/N Code 00 Reserved Reserved 01 500 05 10 600 06 11 Reserved Reserved	TESTER_L OCK
0x440[19:18]	Reserved	2	Reserved	Reserved	Reserved
0x440[20]	Reserved	1	Reserved	Reserved	Reserved
0x440[21]	Reserved	1	Reserved	Reserved	Reserved
0x440[23:22]	Reserved	2	Reserved	Reserved	Reserved
0x440[24]	Reserved	1	Reserved	Reserved	Reserved
0x440[25]	Reserved	1	Reserved	Reserved	Reserved
0x440[26]	Reserved	1	Reserved	Reserved	Reserved
0x440[27]	Reserved	1	Reserved	Reserved	Reserved
0x440[28]	Reserved	1	Reserved	Reserved	Reserved
0x440[29]	Reserved	1	Reserved	Reserved	Reserved
0x440[30]	Reserved	1	Reserved	Reserved	Reserved
0x440[31]	Reserved	1	Reserved	Reserved	Reserved
0x450[7:0]	BOOT_CFG1	8	BOOT configuration register #1, Usage varies, depending on selected boot device.	0x0000 - FlexSPI (NOR) boot 0x01XX - SD boot 0x10XX - MMC/eMMC boot 0x0001 - SEMC (NAND) boot 0x001X - SEMC (NOR) boot 0x10xx - FlexSPI (serial NAND) boot Others - Reserved Refer to Fuse Map for details.	BOOT_CFG_LOCK

Table continues on the next page...

Fusemap Descriptions Table

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x450[15:8]	BOOT_CFG2	8	BOOT configuration register #2, Usage varies, depending on selected boot device.	See fuse-map tab for details.	BOOT_CF G_LOCK
0x450[23:16]	BOOT_CFG3	8	BOOT configuration register #3	See fuse-map tab for details.	BOOT_CF G_LOCK
0x450[31:24]	BOOT_CFG4	8	BOOT configuration register #4	See fuse-map tab for details.	BOOT_CF G_LOCK
0x460[0]	Reserved	1	Reserved	Reserved	Reserved
0x460[1]	SEC_CONFIG[1]	1	Security Configuration (with SEC_CONFIG[0])	00 - FAB (Open) 01 - Open - allows any code to be flashed and executed, even if it has no valid signature. 1x - Closed (Security On)	BOOT_CF G_LOCK
0x460[2]	Boot Frequencies	1	Determines, ARM Core and Bus frequencies during boot	0 - (ARM) 396 / (Bus) 132 MHz 1 - (ARM) 528 / (Bus) 132 MHz	BOOT_CF G_LOCK
0x460[3]	DIR_BT_DIS	1	Direct External Memory Boot Disable	0 - Direct boot from external memory is allowed 1 - Direct boot from external memory is not allowed	BOOT_CF G_LOCK
0x460[4]	BT_FUSE_SEL	1	Determines, whether using fuses for boot configuration, or GPIO / Serial loader	If boot_mode="10" (internal boot) 0=Boot mode configuration is taken from GPIOs. 1=Boot mode configuration is taken from fuses. If boot_mode="00" (boot from fuses) 0 - Boot using Serila Loader (USB) 1- Boot mode configuration is taken from fuses.	BOOT_CF G_LOCK
0x460[5]	FORCE_COLD_BOOT(SBMR)	1	Force cold boot when core comes out of reset. Reflected in SBMR register of SRC	Fuse Function: 0 – Default behavior equivalent to the rest of the product family allowing a fast recovery from low power modes. That is, the ROM is allowed to jump to the address previously programmed in the SRC persistent register. 1 – Fast recovery path in the ROM is not allowed and a cold boot is always performed.	BOOT_CF G_LOCK

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				Customers wanting a higher level of security should burn this fuse.	
0x460[6]	Reserved	1	Reserved	Reserved	Reserved
0x460[7]	Reserved	1	Reserved	Reserved	Reserved
0x460[11:8]	SDRAM_CONFIG[7:0]	4	(SDRAM config options)		BOOT_CF G_LOCK
0x460[13:12]	BEE_KEY0_SEL	2	AES key selection for BEE_KEY0	00 - From register 01 - Reserved 10 - From OTPMK[255:128] 11 - From SW-GP2	BOOT_CF G_LOCK
0x460[15:14]	BEE_KEY1_SEL	2	AES key selection for BEE_KEY1	00 - From register 01 - Reserved 10 - From OTPMK[255:128] 11 - From SW-GP2	BOOT_CF G_LOCK
0x460[16]	FORCE_INTERNAL_BOOT	1	Once been blown, the external BT_MODE[1:0] pins will be ignored, BootROM will take Boot Mode as "internal boot".	0 - Boot Mode from BT_MODE pins 1 - BT_MODE pins be ignored, Boot Mode forced to "internal boot"	BOOT_CF G_LOCK
0x460[17]	SDP_DISABLE	1	Disable/Enable serial download support	0 - Serial download supported 1 - No serial download support	BOOT_CF G_LOCK
0x460[18]	SDP_READ_DISABLE	1	Disable/Enable serial download READ_REGISTER command.	0 - SDP READ_REGISTER command is enabled 1 - SDP READ_REGISTER is disabled	BOOT_CF G_LOCK
0x460[19]	DAP_SJC_SWD_SEL	1	Control DAP works in JTAG or SWD mode	0 - DAP works in SWD mode 1 - DAP works in JTAG mode	BOOT_CF G_LOCK
0x460[20]	SJC_DISABLE	1	Disable/Enable the Secure JTAG Controller module. This fuse is used to create highest JTAG security level, where JTAG is totally blocked.	0 - Secure JTAG Controller is enabled 1 - Secure JTAG Controller is disabled	BOOT_CF G_LOCK
0x460[21]	WDOG_ENABLE	1	Watchdog Enable	Used to specify whether to enable / not watchdog at boot. '0' - Watch-Dog is disabled. '1' - Watch-Dog is enabled.	BOOT_CF G_LOCK
0x460[23:22]	JTAG_SMODE[1:0]	2	JTAG Security Mode. Controls the security mode of the JTAG debug interface	00 - JTAG enable mode 01 - Secure JTAG mode 11 - No debug mode	BOOT_CF G_LOCK

Table continues on the next page...

Fusemap Descriptions Table

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x460[24]	DLL_ENABLE	1	Controls the enable/disable of the DLL for SD/eMMC boot	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC	BOOT_CF G_LOCK
0x460[25]	NAND_ECC_DISABLE	1	Indicate whether to enabled ECC (done by SW/Device HW) for Raw Nand device	0 - NAND ECC is enabled 0 - NAND ECC is disabled	BOOT_CF G_LOCK
0x460[26]	KTE	1	Kill Trace Enable. Enables tracing capability on ETM, and other modules	0 - Bus tracing is allowed 1 - Bus tracing is allowed in case security state as defined by Secure JTAG allows it (for example, JTAG_ENABLE or NO_DEBUG)	BOOT_CF G_LOCK
0x460[27]	JTAG_HEO	1	JTAG HAB Enable Override. Disallows HAB JTAG enabling. The HAB may normally enable JTAG debugging by means of the HAB_JDE-bit in the OCOTP SCS register. The JTAG_HEO-bit can override this behavior	0 - HAB may enable JTAG debug access 1 - HAB JTAG enable is overridden (HAB may not enable JTAG debug access)	BOOT_CF G_LOCK
0x460[28]	Reserved	1			BOOT_CF G_LOCK
0x460[29]	PWR_STABLE_CYCLE_SELECTION	1	Select Power Stable Cycle	0 - 5ms 1 - 2.5ms	BOOT_CF G_LOCK
0x460[31:30]	SD_PWR_CYCLE_SELECTION	2	Select SD Power Cycle	00 - 20ms 01 - 10ms 10 - 5ms 11 - 2.5ms	BOOT_CF G_LOCK
0x470[0]	Override SD Pad Settings	1	Overrides ROM default value for SD PAD control register. When set ROM will override SD pad control register with value programmed into PAD_SETTINGS fuse bits.		BOOT_CF G_LOCK
0x470[1]	BT_MPUM_DISABLE	1	The fuse bit is used for ROM to not enable MMU	0 - MPU is enabled during boot 1 - MPU is disabled during boot	BOOT_CF G_LOCK
0x470[2]	L1 I-Cache DISABLE	1	The fuse bit is used for ROM to not enable L1 I-Cache	0 - L1 I-Cache is enabled during boot 1 - L1 I-Cache is disabled during boot	BOOT_CF G_LOCK

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x470[3]	Disable SDMMC Manufacture mode	1	The fuse bit is used to disable ROM feature "SD/MMC Manufacturing Mode".	0 - Enable 1 – Disable	BOOT_CF G_LOCK
0x470[4]	UART Serial Download Disable	1	Disable UART Serial Download	0 - Disable 1 - Enable	BOOT_CF G_LOCK
0x470[5]	SD2 VOLTAGE SELECTION	1	Fuse bit to change voltage selection for SD2 pads. When set ROM will select 1.8V for SD3 pads otherwise 3.3V	0 - 3.3V 1 - 1.8V	BOOT_CF G_LOCK
0x470[6]	SD1_RST_POLARITY_SELECT	1	Select reset polarity for SD1	0 - Reset active low 1 - Reset active high	BOOT_CF G_LOCK
0x470[7]	DLL Override	1	Select the DLL mode for SD/eMMC.	0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	BOOT_CF G_LOCK
0x470[8]	USDHC IOMUX SRE Enable	1	The fuse bit is used for ROM to enable SRE bit for SD pads	0 - Disable 1 - Enable	BOOT_CF G_LOCK
0x470[9]	USDHC_IOMUX_SION_BIT_ENABLE	1	The fuse bit is used for ROM to enable SION bit for MUX control register.	0 - Disable 1 - Enable	BOOT_CF G_LOCK
0x470[10]	Boot Failure Indicator Pin Select[4]	1	Indicate Boot Failure	00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default	BOOT_CF G_LOCK
0x470[11]	ENABLE_EMMC_22K_PULLUP	1	The fuse bit is used for ROM to enable 22K pullup for SD pads.	0 - 47K pullup 1 - 22K pullup	BOOT_CF G_LOCK
0x470[12]	USDHC_PAD_PULL_DOWN	1	The fuse bit is used for ROM to enable pull down bit for SD pads.	0 - no action 1 - pull down	BOOT_CF G_LOCK
0x470[13]	Override HYS bit for SD/MMC pads	1	Once this fuse be blown, the [HYS] bit of IOMUXC_SW_PAD_CTL_PAD_SDx_CLK(x is the SD port which the ROM boot from), IOMUXC_SW_PAD_CTL_PAD_SDx_CMD, IOMUXC_SW_PAD_CTL_PAD_SDx_DAT0-n(n will be 0, 3, or 7,		BOOT_CF G_LOCK

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
			depends on the bus width selected) will be set.		
0x470[14]	eMMC 4.4 - RESET TO PRE-IDLE STATE	1	Once this fuse be blown, the CMD0 with argument 0xf0f0f0f0 will be sent to put the eMMC card into pre-IDLE state so that eMMC card's fast boot can work properly. This is useful for the warm boot, such as boot due to the WDOG reset.		BOOT_CF G_LOCK
0x470[15]	SD2_RST_POLARITY_SELECT	1	Select reset polarity for SD2	0 - Reset active low 1 - Reset active high	BOOT_CF G_LOCK
0x470[19:16]	Boot Failure Indicator Pin Select[3:0]	4	Misscellanious power management configuration bits.	00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default	BOOT_CF G_LOCK
0x470[20]	BT_LP_B_POLARITY	1	Define GPIO3 polarity, for determining LPB boot mode.	0' - Active High '1' -Active Low	BOOT_CF G_LOCK
0x470[22:21]	LPB_BOOT	2	Divide (Core / Bus) based on Boot Frequencies	'00' - Div by 1; '01' - Div by 2; '10' - Div by 4; '11' - Div by 8	BOOT_CF G_LOCK
0x470[23]	Reserved	1	Reserved	Reserved	Reserved
0x470[30:24]	MMC_DLL_DLY[6:0]	7	eMMC 4.4 delay line default value (set by boot rom), used in conjunction with "DLL Override" = 1 (BOOT_CFG3[3])	Conneted to LVDS module.	BOOT_CF G_LOCK
0x470[31]	Override NAND Pad Settings	1	Override pad settings for NAND boot.		BOOT_CF G_LOCK
0x480[5:0]	Reserved	6	Reserved	Reserved	Reserved
0x480[7:6]	Reserved	2	Reserved	Reserved	Reserved
0x480[15:8]	Reserved	8			MEM_TRIM_LOCK
0x480[23:16]	Reserved	8	Reserved	Reserved	Reserved
0x480[26:24]	Reserved	3	Reserved	Reserved	Reserved
0x480[27]	Reserved	1	Reserved	Reserved	Reserved

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x480[29:28]	Reserved	2	Reserved	Reserved	Reserved
0x480[30]	Reserved	1	Reserved	Reserved	Reserved
0x480[31]	Reserved	1	Reserved	Reserved	Reserved
0x490[9:0]	Reserved	10	Reserved	Reserved	Reserved
0x490[23:10]	Reserved	14	Reserved	Reserved	Reserved
0x490[27:24]	Reserved	4	Reserved	Reserved	Reserved
0x490[31:28]	Reserved	4	Reserved	Reserved	Reserved
0x4A0[7:0]	Reserved	8	Reserved	Reserved	Reserved
0x4A0[19:8]	Reserved	76	Reserved	Reserved	Reserved
0x4C0[31:20]	Reserved	12	Reserved	Reserved	Reserved
0x4D0[31:0]	Reserved	32	Reserved	Reserved	Reserved
0x4E0[31:0]	Reserved	32	Reserved	Reserved	Reserved
0x4F0[15:0]	USB_VID[31:0]	16	USB VID		ANALOG_LOCK
0x4F0[31:16]	USB_PID[31:0]	16	USB PID		ANALOG_LOCK
0x500[31:0]	Reserved	256	Reserved	Reserved	Reserved
0x580[31:0]	SRK_HASH[255:0]	256	SRK key, no HW visible lines. NO HW Visible signals available		SRK_LOCK
0x600[23:0]	SJC_RESP[55:0]	56	Response reference value for the secure JTAG controller		SJC_RESP_LOCK (locks also for read and explicit sense)
0x610[31:24]	Reserved	8	Reserved	Reserved	Reserved
0x620[15:0]	MAC1_ADDR[47:0]	48	Ethernet MAC Address		MAC_ADDR_LOCK
0x630[31:16]	Reserved	16	Reserved	Reserved	Reserved
0x640[31:0]	GP3[31:0]	32	General Purpose fuse register #3		GP3_LOCK
0x650[31:0]	Reserved	32	Reserved	Reserved	Reserved
0x660[31:0]	GP1[31:0]	32	General Purpose fuse register #1		GP1_LOCK
0x670[31:0]	GP2[31:0]	32	General Purpose fuse register #2		GP2_LOCK
0x680[31:0]	SW_GP1[31:0]	32	SW general purpose key		SW_GP1_LOCK
0x690[31:0]	SW_GP2[127:0]	128	SW general purpose key		SW_GP2_LOCK
0x6D0[5:0]	PAD_SETTINGS	6	Used with conjunction of MMC/SD/Nand "Override"	IO pads settings of selected boot interface, are override with this fuses, as follow:	MISC_CONFIG_LOCK

Table continues on the next page...

Fusemap Descriptions Table

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
			Pad Settings" fuse value, as follow: '0' - Use IO default settings for boot device IO pads. '1' - Use "Override" value, as set by this register.	[0] - Slew Rate [3:1] Drive Strength [5:4] - Speed Settings. Refer to IO PAD chapter for "Settings" fields value	
0x6D0[6]	USB_VBUS_EVENT_HANDLER_EN	1	Rom handle USB VBUS attach/detach event	0 - ROM not handle USB VBUS attach/detach event 1 - ROM handle USB VBUS attach/detach event	MISC_CON_F_LOCK
0x6D0[7]	Enable Boot Failure Indicator Pin	1	Enable Boot Failure Indicator Pin	0 - disabled 1 - enabled	MISC_CON_F_LOCK
0x6D0[11:8]	READ_RETRY_SEQ_ID	4	Choose Read Retry Sequence	0000 - don't use read retry(RR) sequence embedded in ROM 0001 - Micron 20nm RR sequence 0010 - Toshiba A19nm RR sequence 0011 - Toshiba 19nm RR sequence 0100 - SanDisk 19nm RR sequence 0101 - SanDisk 1ynmRR sequence Others - Reserved	MISC_CON_F_LOCK
0x6D0[12]	Reserved	1	Unallocated	Reserved	Reserved
0x6D0[15:13]	WDOG Timeout Select	3	Select WDOG Timeout	000 - 64s 001 - 32s 010 - 16s 011 - 8s 100 - 4s Others - Reserved	MISC_CON_F_LOCK
0x6D0[19:16]	Default_FlexRAM_Part	4	Default FlexRAM RAM bank partitioning	CFG DTCM ITCM ORAM RAM_Bank0-15_CFG (O=ORAM,I=ITCM,D=DTCM) 0000: 128KB 128KB 256KB {O O O O D I I I I D D O O O O} 0001: 128KB 64KB 320KB {O O O D D I I D D O O O O O O} 0010: 128KB 256KB 128KB {O D D I I I I I I D D O O}	MISC_CON_F_LOCK

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				0011: 128KB 32KB 352KB {O O O D D D D I O O O O O O O O} 0100: 64KB 128KB 320KB {O O O O D D I I I I O O O O O O O} 0101: 64KB 64KB 384KB {O O O O D D I I O O O O O O O O O} 0110: 64KB 256KB 192KB {O O D D I I I I I I I I O O O O} 0111: 0KB 442KB 64KB {O O I I I I I I I I I I I I I I I I I I} 1000: 256KB 128KB 128KB {O O D D D D I I I I D D D D O O O} 1001: 256KB 64KB 192KB {O O D D D D I I D D D D O O O O O} 1010: 192KB 256KB 64KB {O O D D I I I I I I D D D D D D D} 1011: 448KB 0KB 64KB {O O D D D D D D D D D D D D D D D} 1100: 0KB 128KB 384KB {O O O O I I I I O O O O O O O O O O O} 1101: 32KB 32KB 448KB {O O O D I O O O O O O O O O O O O O O O} 1110: 0KB 256KB 256KB {O O I I I I I I I I I O O O O O O O O O} 1111: 0KB 0KB 512KB {O O O O O O O O O O O O O O O O O O O O}	
0x6D0[21:20]	Reserved	2	Reserved	Reserved	Reserved
0x6D0[22]	Reserved	1	Reserved	Reserved	Reserved
0x6D0[23]	Reserved	1	Reserved	Reserved	Reserved
0x6D0[24]	EEPROM_RECO VERY_EN	1	EEPROM Recovery Enable	0' - Disabled '1' - Enabled	MISC_CON F_LOCK
0x6D0[26:25]	LPSPI_PORT_SE L	2	LPSPI Port Select	00 - LPSPI1 01 - LPSPI2 10 - LPSPI3 11 - LPSPI4	MISC_CON F_LOCK
0x6D0[27]	LPSPI_ADDR	1	SPI Addressing	0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)	MISC_CON F_LOCK
0x6D0[29:28]	LPSPI_CS_SEL	2	LPSPI CS select	00 - CS#0 (default) 01 - CS#1 10 - CS#2 11 - CS#3	MISC_CON F_LOCK

Table continues on the next page...

Fusemap Descriptions Table

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x6D0[31:30]	SD_CALIBRATION_STEP	2	SD Calibration Step	00' - 1	MISC_CON_F_LOCK
0x6E0[3:0]	OVERRIDE_BUS_Y_OFFSET (SPI NAND BOOT)	4	Override busy offset enable	0' - Disable override '1' - Enable override	MISC_CON_F_LOCK
0x6E0[6:4]	PREAMBLE_DELAY (RAW NAND)	3	RAW NAND Toggle Mode 33MHz Preamble Delay, Read Latency	000' - 16 GPMICLK cycles '001' - 1 GPMICLK cycles '010' - 2 GPMICLK cycles '011' - 3 GPMICLK cycles '100' - 4 GPMICLK cycles '101' - 5 GPMICLK cycles '110' - 6 GPMICLK cycles '111' - 7 GPMICLK cycles	MISC_CON_F_LOCK
0x6E0[7]	BUSY_BIT_OFFSET (SPI NAND BOOT)	1	BUSY BIT OFFSET		MISC_CON_F_LOCK
0x6E0[13:8]	BYPASS_READ_STATUS	6	Bypass read status	0' – Use default read status cmd '1' – Bypass default read status cmd, use PAGE_READ_TIME to wait while executing page read command	MISC_CON_F_LOCK
0x6E0[14]	BYPASS_ECC_READ	1	Bypass ecc read	0' – Use default ECC read cmd '1' – Use the specified ECC read LUT sequences provided by ECC_READ_LUT fuse fields	MISC_CON_F_LOCK
0x6E0[15]	PAGE_READ_TIME (SPI NAND)	1	Page read time		MISC_CON_F_LOCK
0x6E0[23:16]	NAND_READ_CMD_CODE1[7:0]	8	NAND_READ_CMD_CODE1	For RawNAND: First command word to be used for Nand read For SPINAND: Page read command	MISC_CON_F_LOCK
0x6E0[31:24]	NAND_READ_CMD_CODE2[7:0]	8	NAND_READ_CMD_CODE2	For RawNAND: Second command word to be used for Nand read For SPINAND: Cache read command	MISC_CON_F_LOCK
0x6E0[7:0]	BOOT_CONFIG_MISC0	8	boot configuration misc		MISC_CON_F_LOCK
0x6E0[15:8]	BOOT_CONFIG_MISC1	8	boot configuration misc		MISC_CON_F_LOCK
0x6E0[23:16]	BOOT_CONFIG_MISC2	8	boot configuration misc		MISC_CON_F_LOCK
0x6E0[31:24]	BOOT_CONFIG_MISC3	8	boot configuration misc		MISC_CON_F_LOCK
0x6F0[2:0]	Reserved	3	Reserved	Reserved	Reserved
0x6F0[3]	Reserved	1	Reserved	Reserved	Reserved
0x6F0[7:4]	Reserved	4	Reserved	Reserved	Reserved
0x6F0[15:8]	Reserved	8	Reserved	Reserved	Reserved

Table continues on the next page...

Table 4-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x6F0[31:16]	Reserved	16	Reserved	Reserved	Reserved

Chapter 5

Central Security Unit (CSU)

5.1 Overview

The CSU manages the system security policy for peripheral access on the SoC. The CSU allows trusted code to set individual security access privileges on each of the peripherals, using one of eight security access privilege levels. Also, according to programmed policy, the CSU may assign bus master security privileges during bus transactions.

5.1.1 Features

The Central Security Unit (CSU) sets the access control policies between the bus masters and bus slaves, allowing for peripherals to be separated into distinct security domains. This protects against unauthorized access to data, for example, when the software programs a DMA bus master to access the addresses that the software is prohibited from accessing directly. By configuring the DMA bus master privileges in the CSU to be consistent with the software privileges defends against such access attempts. Additionally, the CSU manages the system security alarms. These alarms are signals routed from various SoC peripherals and I/Os that indicate security-violation conditions.

The CSU has these security-related features:

- Peripheral access policy—the appropriate bus master privilege and identity are required to access each peripheral.
- Masters privilege policy—the CSU overrides the bus master privilege signals (user/supervisor, secure/non-secure) according to the access control policy.

5.2 Functional description

The CSU enables the secure software to set the bus privilege security policy within the platform.

The security policies may be set, and optionally locked in the CSU registers. The examples of a secure software include the Command Sequence File (CSF) processed by the High Assurance Boot (HAB) or the HAB-authenticated image which executes after the boot ROM.

5.2.1 Peripheral access policy

According to its programmed policy, the CSU determines the bus master privileges and the masters that are allowed to access each of the slave peripherals.

There are four security modes of operation (bus privileges) in the system distinguished by the security (TrustZone/non-TrustZone) and privilege (Supervisor/User) setting of the module. This is the list of these security modes, organized from the highest security level to the lowest:

- TrustZone (secure) privilege (supervisor) mode—highest security level
- TrustZone (secure) non-privilege (user) mode—medium security level
- Non-TrustZone (regular) privilege (supervisor) mode—medium security level
- Non-TrustZone (regular) non-privilege (user) mode—lowest security level

This functionality is implemented as follows:

The Configure Slave Level (CSL) Register value for a specified peripheral resource defines the output signal (csu_sec_level) for that peripheral. The value of this signal determines the master privileges that can access the peripheral. The relationship between the value of the csu_sec_level signal and the security operation mode is shown in this table:

Table 5-1. Access permissions

CSU_SEC_LEVEL[2:0]	Non-secure user mode	Non-secure SPVR mode	Secure (TZ) user mode	Secure (TZ) SPVR mode	CSL register value
(0) 000	RD+WR	RD+WR	RD+WR	RD+WR	8'b1111_1111
(1) 001	None	RD+WR	RD+WR	RD+WR	8'b1011_1011
(2) 010	RD	RD	RD+WR	RD+WR	8'b0011_1111
(3) 011	None	RD	RD+WR	RD+WR	8'b0011_1011
(4) 100	None	None	RD+WR	RD+WR	8'b0011_0011
(5) 101	None	None	None	RD+WR	8'b0010_0010
(6) 110	None	None	RD	RD	8'b0000_0011
(7) 111	None	None	None	None	Any other value

5.2.2 Initialization policy

The recommended initialization procedure is as follows:

1. Write the CSU_CSL register field value to indicate each peripheral's privilege mode.
2. Write the HP register field value to override the master's privilege mode.

NOTE

After programming, the register lock bit must be set to prevent further modifications to a register value.

5.3 Programmable Registers

The following sections provide a detailed description of the CSU registers and their respective bit and field assignments. Assume that the base address is 021C.

- The CSU registers: CSU_CSL, CSU_HP, CSU_SA , and CSU_HPCONTROL can only be written in the secure supervisor mode. (Note: These registers are also referred to as the security control registers (or SCRs) in this document)
- The previous cycle's lock bit is checked while writing to a register. If the lock bit was cleared in the previous cycle and is being set during the current cycle, then the register fields covered by that lock bit may be written during the current cycle.

CSU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_C000	Config security level register (CSU_CSL0)	32	R/W	0033_0033h	5.3.1/178
400D_C004	Config security level register (CSU_CSL1)	32	R/W	0033_0033h	5.3.1/178
400D_C008	Config security level register (CSU_CSL2)	32	R/W	0033_0033h	5.3.1/178
400D_C00C	Config security level register (CSU_CSL3)	32	R/W	0033_0033h	5.3.1/178
400D_C010	Config security level register (CSU_CSL4)	32	R/W	0033_0033h	5.3.1/178
400D_C014	Config security level register (CSU_CSL5)	32	R/W	0033_0033h	5.3.1/178
400D_C018	Config security level register (CSU_CSL6)	32	R/W	0033_0033h	5.3.1/178
400D_C01C	Config security level register (CSU_CSL7)	32	R/W	0033_0033h	5.3.1/178
400D_C020	Config security level register (CSU_CSL8)	32	R/W	0033_0033h	5.3.1/178
400D_C024	Config security level register (CSU_CSL9)	32	R/W	0033_0033h	5.3.1/178
400D_C028	Config security level register (CSU_CSL10)	32	R/W	0033_0033h	5.3.1/178
400D_C02C	Config security level register (CSU_CSL11)	32	R/W	0033_0033h	5.3.1/178
400D_C030	Config security level register (CSU_CSL12)	32	R/W	0033_0033h	5.3.1/178

Table continues on the next page...

CSU memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_C034	Config security level register (CSU CSL13)	32	R/W	0033_0033h	5.3.1/178
400D_C038	Config security level register (CSU CSL14)	32	R/W	0033_0033h	5.3.1/178
400D_C03C	Config security level register (CSU CSL15)	32	R/W	0033_0033h	5.3.1/178
400D_C040	Config security level register (CSU CSL16)	32	R/W	0033_0033h	5.3.1/178
400D_C044	Config security level register (CSU CSL17)	32	R/W	0033_0033h	5.3.1/178
400D_C048	Config security level register (CSU CSL18)	32	R/W	0033_0033h	5.3.1/178
400D_C04C	Config security level register (CSU CSL19)	32	R/W	0033_0033h	5.3.1/178
400D_C050	Config security level register (CSU CSL20)	32	R/W	0033_0033h	5.3.1/178
400D_C054	Config security level register (CSU CSL21)	32	R/W	0033_0033h	5.3.1/178
400D_C058	Config security level register (CSU CSL22)	32	R/W	0033_0033h	5.3.1/178
400D_C05C	Config security level register (CSU CSL23)	32	R/W	0033_0033h	5.3.1/178
400D_C060	Config security level register (CSU CSL24)	32	R/W	0033_0033h	5.3.1/178
400D_C064	Config security level register (CSU CSL25)	32	R/W	0033_0033h	5.3.1/178
400D_C068	Config security level register (CSU CSL26)	32	R/W	0033_0033h	5.3.1/178
400D_C06C	Config security level register (CSU CSL27)	32	R/W	0033_0033h	5.3.1/178
400D_C070	Config security level register (CSU CSL28)	32	R/W	0033_0033h	5.3.1/178
400D_C074	Config security level register (CSU CSL29)	32	R/W	0033_0033h	5.3.1/178
400D_C078	Config security level register (CSU CSL30)	32	R/W	0033_0033h	5.3.1/178
400D_C07C	Config security level register (CSU CSL31)	32	R/W	0033_0033h	5.3.1/178
400D_C200	HP0 register (CSU_HPO)	32	R/W	0000_0000h	5.3.2/182
400D_C218	Secure access register (CSU_SA)	32	R/W	0000_0000h	5.3.3/185
400D_C358	HPCONTROL0 register (CSU_HPCONTROL0)	32	R/W	0000_0000h	5.3.4/188

5.3.1 Config security level register (CSU_CSLn)

There are several config security level (CSU_CSL0-CSU_CSLn) registers. Each CSU_CSL comprises of two fields, with each field used to determine the read and write access permissions for a slave peripheral. These 8-bit fields for the first and second slaves are located in b23-b16 and bits b7-b0, respectively.

The permission access table [Table 5-1](#) shows the security levels and the csu_sec_level signal levels corresponding to different values of the 8-bit CSU_CSL field for a given slave.

Most slaves have unique CSL registers. Some slaves are grouped together in USB, Timers, PowerUp, and Audio groups. The following table shows the allocation of the CSL register per slave or a group of slave modules.

Table 5-2. CSL slave modules mapping

Slave module	Corresponding CSL register and bit field	Comments
CCM	CSL0[7:0]	
SRC	CSL0[23:16]	
GPC	CSL1[7:0]	
Reserved	CSL1[23:16]	
DMA_CH_MUX	CSL2[7:0]	
EDMA	CSL2[23:16]	
SJC	CSL3[7:0]	
TSC_DIG	CSL3[23:16]	
CSU	CSL4[7:0]	
ANATOP	CSL4[23:16]	
SNVS_HP	CSL5[7:0]	
RT_WDOG	CSL5[23:16]	
TRNG	CSL6[7:0]	
ADC1 ADC2	CSL6[23:16]	
WDOG1 WDOG3 EWM	CSL7[7:0]	
FLEXRAM	CSL7[23:16]	
ACMP	CSL8[7:0]	
PIT	CSL8[23:16]	
DCDC	CSL9[7:0]	
KPP	CSL9[23:16]	
IOMUXC_IOMUXC_GPR	CSL10[7:0]	
OCOTP	CSL10[23:16]	
GPT1 GPT2	CSL11[7:0]	
QTimer1 QTimer2 QTimer3 QTimer4	CSL11[23:16]	
CAN1 CAN2	CSL12[7:0]	
GPIO1 GPIO2 GPIO3 GPIO4	CSL12[23:16]	
FlexIO1 FlexIO2	CSL13[7:0]	
LPUART1 LPUART2 LPUART3 LPUART4 LPUART5 LPUART6 LPUART7 LPUART8	CSL13[23:16]	
ROMCP	CSL14[7:0]	
DCP	CSL14[23:16]	
SEMC	CSL15[7:0]	
USB	CSL15[23:16]	
ENET	CSL16[7:0]	
USDHC1 USDHC2	CSL16[23:16]	
PXP LCDIF CSI	CSL17[7:0]	
FlexSPI	CSL17[23:16]	
LPI2C3 LPI2C4	CSL18[7:0]	

Table continues on the next page...

Table 5-2. CSL slave modules mapping (continued)

Slave module	Corresponding CSL register and bit field	Comments
LPI2C1 LPI2C2	CSL18[23:16]	
BEE	CSL19[7:0]	
FLEXPWM1 FLEXPWM2 FLEXPWM3 FLEXPWM4	CSL19[23:16]	
Reserved	CSL20[7:0]	
ENC1 ENC2 ENC3 ENC4	CSL20[23:16]	
AOI1 AOI2 XBAR1 XBAR2 XBAR3	CSL21[7:0]	
ADC_ETC	CSL21[23:16]	
Reserved	CSL22[7:0]	
LPSPI1 LPSPI2 LPSPI3 LPSPI4	CSL22[23:16]	
SAI1 SAI2 SAI3 SPDIF	CSL23[7:0]	
IOMUXC_SNVS IOMUXC_SNVS_GPR	CSL23[23:16]	
Reserved	CSL24 to CSL31	

Address: 400D_C000h base + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	Reserved							LOCK_S1		NSW_S1	NUW_S1	SSW_S1	SUW_S1	NSR_S1	NUR_S1	SSR_S1	SUR_S1
W										0	0	1	1	0	0	1	1
Reset	0	0	0	0	0	0	0	0		0	0	1	1	0	0	1	1
Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	Reserved							LOCK_S2		NSW_S2	NUW_S2	SSW_S2	SUW_S2	NSR_S2	NUR_S2	SSR_S2	SUR_S2
W										0	0	1	1	0	0	1	1
Reset	0	0	0	0	0	0	0	0		0	0	1	1	0	0	1	1

CSU_CSLn field descriptions

Field	Description
31–25 -	This field is reserved. Reserved.
24 LOCK_S1	The lock bit corresponding to the first slave. It is written by the secure software. 0 Not locked. The bits 16-23 can be written by the software. 1 The bits 16-23 are locked and can't be written by the software.
23 NSW_S1	Non-secure supervisor write access control for the first slave 0 The non-secure supervisor write access is disabled for the first slave. 1 The non-secure supervisor write access is enabled for the first slave

Table continues on the next page...

CSU_CSLn field descriptions (continued)

Field	Description
22 NUW_S1	Non-secure user write access control for the first slave 0 The non-secure user write access is disabled for the first slave. 1 The non-secure user write access is enabled for the first slave.
21 SSW_S1	Secure supervisor write access control for the first slave 0 The secure supervisor write access is disabled for the first slave. 1 The secure supervisor write access is enabled for the first slave.
20 SUW_S1	Secure user write access control for the first slave 0 The secure user write access is disabled for the first slave. 1 The secure user write access is enabled for the first slave.
19 NSR_S1	Non-secure supervisor read access control for the first slave 0 The non-secure supervisor read access is disabled for the first slave. 1 The non-secure supervisor read access is enabled for the first slave.
18 NUR_S1	Non-secure user read access control for the first slave 0 The non-secure user read access is disabled for the first slave. 1 The non-secure user read access is enabled for the first slave.
17 SSR_S1	Secure supervisor read access control for the first slave 0 The secure supervisor read access is disabled for the first slave. 1 The secure supervisor read access is enabled for the first slave.
16 SUR_S1	Secure user read access control for the first slave 0 The secure user read access is disabled for the first slave. 1 The secure user read access is enabled for the first slave.
15–9 -	This field is reserved. Reserved
8 LOCK_S2	The lock bit corresponding to the second slave. It is written by the secure software. 0 Not locked. Bits 7-0 can be written by the software. 1 Bits 7-0 are locked and cannot be written by the software
7 NSW_S2	Non-secure supervisor write access control for the second slave 0 The non-secure supervisor write access is disabled for the second slave. 1 The non-secure supervisor write access is enabled for the second slave.
6 NUW_S2	Non-secure user write access control for the second slave 0 The non-secure user write access is disabled for the second slave. 1 The non-secure user write access is enabled for the second slave.
5 SSW_S2	Secure supervisor write access control for the second slave 0 The secure supervisor write access is disabled for the second slave. 1 The secure supervisor write access is enabled for the second slave.
4 SUW_S2	Secure user write access control for the second slave

Table continues on the next page...

CSU_CSLn field descriptions (continued)

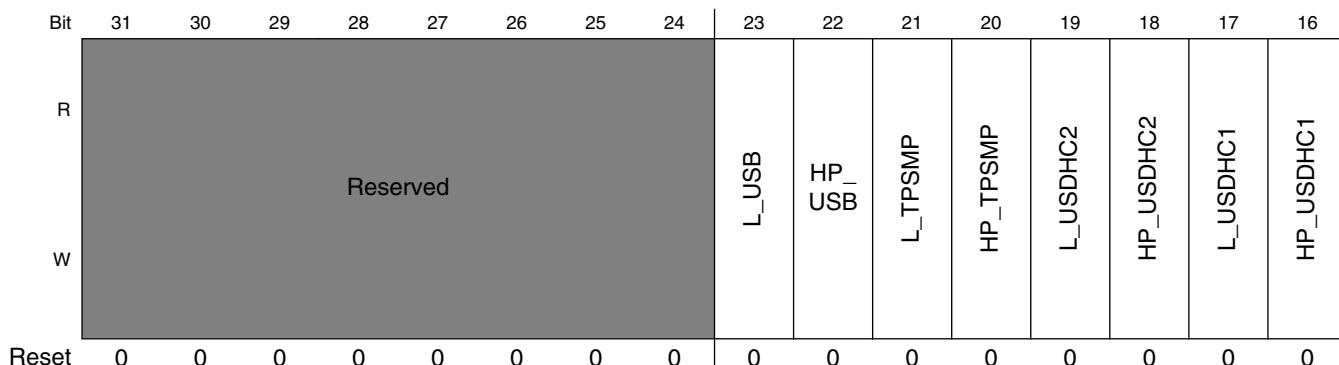
Field	Description
	0 The secure user write access is disabled for the second slave. 1 The secure user write access is enabled for the second slave.
3 NSR_S2	Non-secure supervisor read access control for the second slave 0 The non-secure supervisor read access is disabled for the second slave. 1 The non-secure supervisor read access is enabled for the second slave.
2 NUR_S2	Non-secure user read access control for the second slave 0 The non-secure user read access is disabled for the second slave. 1 The non-secure user read access is enabled for the second slave.
1 SSR_S2	Secure supervisor read access control for the second slave 0 The secure supervisor read access is disabled for the second slave. 1 The secure supervisor read access is enabled for the second slave.
0 SUR_S2	Secure user read access control for the second slave 0 The secure user read access is disabled for the second slave. 1 The secure user read access is enabled for the second slave.

5.3.2 HP0 register (CSU_HP0)

The CSU_HP0 register can be programmed to determine the privilege (either the user mode or the supervisor mode) for several different master groups. The privilege of a particular master group can be overridden by muxing it with the corresponding bit in this register.

The even bit positions (CSU_HP0[30,28,...0]) in the register hold the privilege indicator bits; while the odd bit positions (CSU_HP0[31,29,...,1]) contain lock bits which enable/disable writing to the corresponding privilege indicator bits.

Address: 400D_C000h base + 200h offset = 400D_C200h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L_ENET	HP_ENET	Reserved	Reserved	L_DCP	HP_DCP	L_PXP	HP_PXP	L_CSI	HP_CSI	L_LCDIF	HP_LCDIF	L_DMA	HP_DMA	Reserved	Reserved
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSU_HP0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 L_USB	Lock bit set by the TZ software for the USB 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
22 HP_USB	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the USB. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
21 L_TPSMP	Lock bit set by the TZ software for the TPSMP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
20 HP_TPSMP	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the TPSMP. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
19 L_USDHC2	Lock bit set by the TZ software for the USDHC2 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
18 HP_USDHC2	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the USDHC2. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
17 L_USDHC1	Lock bit set by the TZ software for the USDHC1 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
16 HP_USDHC1	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the USDHC1. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.

Table continues on the next page...

CSU_HP0 field descriptions (continued)

Field	Description
15 L_ENET	Lock bit set by the TZ software for the ENET 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
14 HP_ENET	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the ENET. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
13 -	This field is reserved. Reserved
12 -	This field is reserved. Reserved
11 L_DCP	Lock bit set by the TZ software for the DCP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit cannot be written by the software.
10 HP_DCP	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the DCP. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
9 L_PXP	Lock bit set by the TZ software for the PXP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
8 HP_PXP	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the PXP. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
7 L_CSI	Lock bit set by the TZ software for the CSI 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
6 HP_CSI	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the CSI. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
5 L_LCDIF	Lock bit set by the TZ software for the LCDIF 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
4 HP_LCDIF	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the LCDIF. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
3 L_DMA	Lock bit set by the TZ software for the eDMA

Table continues on the next page...

CSU_HP0 field descriptions (continued)

Field	Description
	0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
2 HP_DMA	Determines whether the register value of the corresponding HP field is passed as the hprot[1] of the eDMA. 0 The hprot1 input signal value is routed to the csu_hprot1 output for the corresponding master. 1 The HP register bit is routed to the csu_hprot1 output for the corresponding master.
1 -	This field is reserved. Reserved
0 -	This field is reserved. Reserved

5.3.3 Secure access register (CSU_SA)

The secure access register can be programmed to specify the access policy (either secure or non-secure) for up to several different masters. This register is used to set the access policy for the type-1 masters which are not capable of setting the policy by themselves.

The 16 even bit positions (CSU_SA[30,28,...,0]) in the register hold the policy indicator bits, while the odd bit positions (CSU_SA[31,29,...,1]) contain the lock bits which enable/disable writing to the corresponding policy indicator bits.

Address: 400D_C000h base + 218h offset = 400D_C218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved									L_USB	NSA_USB	L_TPSMP	NSA_TPSMP	L_USDHC2	NSA_USDHC2	L_USDHC1	NSA_USDHC1
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	L_ENET	NSA_ENET	Reserved	Reserved	L_DCP	NSA_DCP	L_PXP	NSA_PXP	L_CS1	NSA_CS1	L_LCDIF	NSA_LCDIF	L_DMA	NSA_DMA	Reserved	Reserved	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CSU_SA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23 L_USB	Lock bit set by the TZ software for the USB 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
22 NSA_USB	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the USB 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
21 L_TPSMP	Lock bit set by the TZ software for the TPSMP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
20 NSA_TPSMP	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the TPSMP 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
19 L_USDHC2	Lock bit set by the TZ software for the USDHC2 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
18 NSA_USDHC2	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) Access for the USDHC2 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
17 L_USDHC1	Lock bit set by the TZ software for the USDHC1 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
16 NSA_USDHC1	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the USDHC1 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
15 L_ENET	Lock bit set by the TZ software for the ENET1 and ENET2 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
14 NSA_ENET	Non-secure access policy indicator bit Access policy indicator bit for the ENET1 and ENET2 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master

Table continues on the next page...

CSU_SA field descriptions (continued)

Field	Description
13 -	This field is reserved. Reserved
12 -	This field is reserved. Reserved
11 L_DCP	Lock bit set by the TZ software for the DCP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
10 NSA_DCP	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the DCP 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
9 L_PXP	Lock bit set by the TZ software for the PXP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
8 NSA_PXP	Non-Secure Access Policy indicator bit Indicates the access type (secure/non-Secure) for the PXP 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
7 L_CSI	Lock bit set by the TZ software for the CSI 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
6 NSA_CSI	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the CSI 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
5 L_LCDIF	Lock bit set by the TZ software for the LCDIF 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
4 NSA_LCDIF	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the LCDIF 0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
3 L_DMA	Lock bit set by the TZ software for the eDMA 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
2 NSA_DMA	Non-secure access policy indicator bit Indicates the access type (secure/non-secure) for the eDMA

Table continues on the next page...

CSU_SA field descriptions (continued)

Field	Description
	0 Secure access for the corresponding type-1 master 1 Non-secure access for the corresponding type-1 master
1 -	This field is reserved. Reserved
0 -	This field is reserved. Reserved

5.3.4 HPCONTROL0 register (CSU_HPCONTROL0)

The HP control register CSU_HPCONTROL0 enables the CSU to control the USER/SUPERVISOR mode state for the specified masters. The register toggles the csu_hprot1 output signal for the system masters. The two possible sources for the csu_hprot1 output are:

1. The hprot1 input signal
2. The corresponding bit in the HP register

The even bits in the registers are used to lock the control bit values.

Address: 400D_C000h base + 358h offset = 400D_C358h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	R								L_USB	HPC_USB	L_TPSMP	HPC_TPSMP	L_USDH2	HPC_USDH2	L_USDH1	HPC_USDH1	
	W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	R	L_ENET	HPC_ENET	Reserved	Reserved	L_DCP	HPC_DCP	L_PXP	HPC_PXP	L_CS1	HPC_CS1	L_LCDIF	HPC_LCDIF	L_DMA	HPC_DMA	Reserved	Reserved
	W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CSU_HPCONTROL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 L_USB	Lock bit set by the TZ software for the USB. 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
22 HPC_USB	Indicates the privilege/user mode for the USB 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
21 L_TPSMP	Lock bit set by the TZ software for the TPSMP. 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
20 HPC_TPSMP	Indicates the privilege/user mode for the TPSMP 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
19 L_USDHC2	Lock bit set by the TZ software for the USDHC2. 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
18 HPC_USDHC2	Indicates the privilege/user mode for the USDHC2 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
17 L_USDHC1	Lock bit set by the TZ software for the USDHC1 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
16 HPC_USDHC1	Indicates the privilege/user mode for the USDHC1 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
15 L_ENET	Lock bit set by the TZ software for the ENET 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
14 HPC_ENET	Indicates the privilege/user mode for the ENET 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
13 -	This field is reserved. Reserved
12 -	This field is reserved. Reserved
11 L_DCP	Lock bit set by the TZ software for the DCP

Table continues on the next page...

CSU_HPCONTROL0 field descriptions (continued)

Field	Description
	0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
10 HPC_DCP	Indicates the privilege/user mode for the DCP 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
9 L_PXP	Lock bit set by the TZ software for the PXP 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
8 HPC_PXP	Indicates the privilege/user mode for the PXP 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
7 L_CSI	Lock bit set by the TZ software for the CSI 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
6 HPC_CSI	Indicates the privilege/user mode for the CSI 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
5 L_LCDIF	Lock bit set by the TZ software for the LCDIF 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
4 HPC_LCDIF	Indicates the privilege/user mode for the LCDIF 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
3 L_DMA	Lock bit set by the TZ software for the eDMA 0 No lock—the adjacent (next lower) bit can be written by the software. 1 Lock—the adjacent (next lower) bit can't be written by the software.
2 HPC_DMA	Indicates the privilege/user mode for the eDMA 0 User mode for the corresponding master 1 Supervisor mode for the corresponding master
1 -	This field is reserved. Reserved
0 -	This field is reserved. Reserved

Chapter 6

Bus Encryption Engine (BEE)

6.1 Introduction

This document describes the micro-architecture for Bus Encryption Engine (BEE). It also describes the block level functionality of BEE.

The BEE module is implemented as an on-the-fly decryption engine. Main features of the BEE module are:

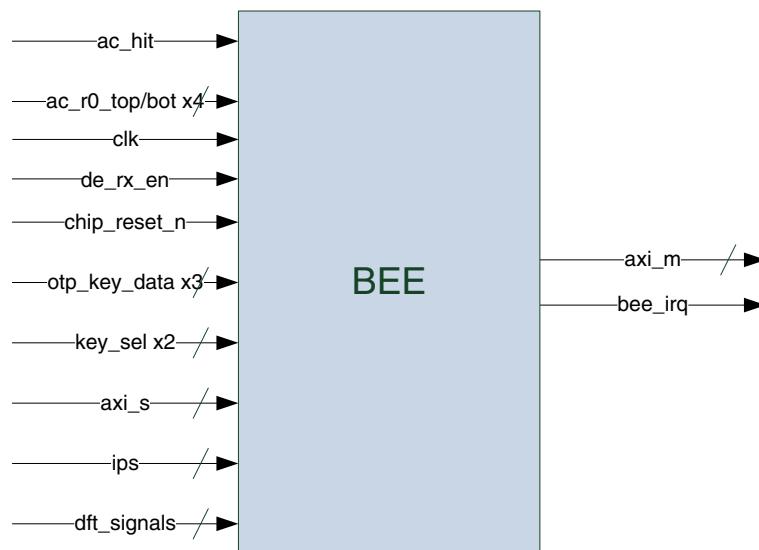
- Standard AXI interconnection
- On-the-fly AES-128 decryption, supporting ECB and CTR mode
- Aliased memory space support. Address remapping for up to two individual regions
- Independent AES Key management for those two individual regions
- Bus access pattern optimization with the aid of local store and forward buffer
- Non-secured access filtering based on security label of the access
- Illegal access check and filtering.

The known hardware limitations of the BEE module are as follows:

- Only supports 128 bits data width AXI interconnection
- Only supports 16-byte burst access size. For single transaction, the minimum supported access size is limited to 4-byte.
- Granularity of the address bias is 128 KB per step
- Maximum supported burst length is limited to 4.

6.1.1 Interface and signal list

The following diagram shows the interface of BEE module. Detail descriptions of the signals are listed in the table below.

**Figure 6-1. BEE Interface****Table 6-1. BEE external control signal list**

Signal Name	Description	Direction	Width	Notes
otp_key_data0	128 bits key required by AES encryption engine	input	[127 : 0]	From eFuse OTPMK[127:0]
otp_key_data1	128 bits key required by AES encryption engine	input	[127 : 0]	From eFuse OTPMK[255:128]
otp_key_data2	128 bits key required by AES encryption engine	input	[127 : 0]	From eFuse SW_GP2[127:0]
otp_key_sel0	AES key selection control for memory region0	Input	[1 : 0]	From eFuse BEE_KEY0_SEL[1:0] 2'b00: from BEE register configuration 2'b01: Reserved 2'b10: OTPMK[255:128] 2'b11: SW_GP2
otp_key_sel1	AES key selection control for memory region1	Input	[1 : 0]	From eFuse BEE_KEY1_SEL[1:0] 2'b00: from BEE register configuration 2'b01: Reserved 2'b10: OTPMK[255:128] 2'b11: SW_GP2
ac_r0_top	End address of access protected region-0	input	[31: 3]	From IOMUXC_GPR_GPR19, M7_APAC_AC_R0_TOP[31:3]

Table continues on the next page...

Table 6-1. BEE external control signal list (continued)

Signal Name	Description	Direction	Width	Notes
ac_r0_bot	Start address of access protected region-0	input	[31: 3]	From IOMUXC_GPR_GPR18 , M7_APAC_AC_R0_BOT[31:3]
ac_r1_top	End address of access protected region-1	input	[31: 3]	From IOMUXC_GPR_GPR21 , M7_APAC_AC_R1_TOP[31:3]
ac_r1_bot	Start address of access protected region-1	input	[31: 3]	From IOMUXC_GPR_GPR20 , M7_APAC_AC_R1_BOT[31:3]
ac_r2_top	End address of access protected region-2	input	[31: 3]	From IOMUXC_GPR_GPR23 , M7_APAC_AC_R2_TOP[31:3]
ac_r2_bot	Start address of access protected region-2	input	[31: 3]	From IOMUXC_GPR_GPR22 , M7_APAC_AC_R2_BOT[31:3]
ac_r3_top	End address of access protected region-3	input	[31: 3]	From IOMUXC_GPR_GPR25 , M7_APAC_AC_R3_TOP[31:3]
ac_r3_bot	Start address of access protected region-3	input	[31: 3]	From IOMUXC_GPR_GPR24 , M7_APAC_AC_R3_BOT[31:3]
de_rx_en	Decryption enable control for those four protected regions.	Input	[3: 0]	From IOMUXC_GPR_GPR11 .BEE_DE_RX_EN <ul style="list-style-type: none"> • de_rx_en[0]. value 0: disable decryption for region-0; value 1: enable decryption for region-0 • de_rx_en[1] for region-1 • de_rx_en[2] for region-2 • de_rx_en[3] for region-3

6.2 BEE u-architecture

The BEE is composed of four main functional sub-blocks: BEE_CTRL, CVC, DE, and IPS_LOGIC, as shown in the following figure.

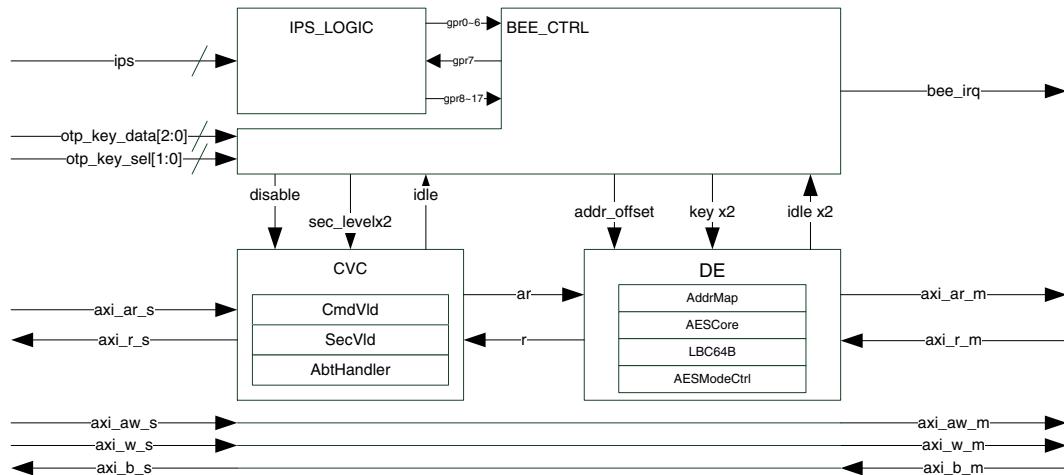


Figure 6-2. BEE u-architecture

6.2.1 BEE_CTRL

The BEE_CTRL block is responsible for:

- Synchronizing external hardwire inputs to BEE's internal domain
- Work mode and internal workflow control
- Internal interrupt handling

6.2.2 Command Validity Checker (CVC)

The CVC block is composed of three functional sub-blocks: CmdVld, SecVld, and AbtHandler.

Command validation (CmdVld)

As BEE only supports 16 byte aligned and full size accesses, the Masters which issue AXI commands to BEE must make sure that all AXI commands do not violate the rule. If an unexpected command is sent to the BEE, CmdVld block will detect the violation and notify AbtHandler of the violation event.

Security level validation (SecVld)

The SecVld block grants access permission to each command based on:

- Security level input (refer to the BEE Internal Registers section for more details)
- Security label (arprot field) of the AXI command. The access permission is granted to each command as shown in the following table¹.

Security level	Non-Secure User Mode	Non-Secure Spvr Mode	Secure (TZ) User Mode	Secure (TZ) Spvr Mode
(0) 00	RD+WR	RD+WR	RD+WR	RD+WR
(1) 01	None	RD+WR	RD+WR	RD+WR
(2) 10	None	None	RD+WR	RD+WR
(3) 11	None	None	None	RD+WR

If an unauthorized command is sent to the BEE, SecVld blocks the command and notifies AbtHandler of the security violation.

Abort handler (AbtHandler)

The AbtHandler is responsible for:

- Collecting abnormal events reported by CmdVld and SecVld
- Illegal command filtering (so illegal commands won't be propagated to the output of BEE)
- Error response generation for detected illegal commands
- Access permission control based on external input singles
 - Access permission is prohibited when:
 - Read access to memory region defined by “ac_r[x]_bot” and “ac_r[x]_top”, and
 - Corresponding ac_hit[x] signal (which is generated by comparing the current CPU instruction fetching address and the boundary of the access protected regions) input is low, and
 - Register configuration gpr0.ac_prot_en is high.
 - Access is granted when:
 - Read access to memory region outside of region defined by “ac_r[x]_bot” and “ac_r[x]_top”, or
 - Read access to memory region defined by “ac_r[x]_bot” and “ac_r[x]_top” and corresponding ac_hit[x] signal input is high, or
 - Register configuration gpr0.ac_prot_en is low.

1. In the table the “Security level” is the security level input; the arprot value corresponding to those 4 modes “Non-Secure User Mode, Non-Secure Spvr Mode, Secure (TZ) User Mode, Secure (TZ) Spvr Mode” is “3’bx00, 3’bx01, 3’bx10, 3’bx11”.

6.2.3 Decryption Engine (DE)

The Decryption Engine is composed of four functional sub-blocks: AddrMap, AESCore, LBC64B, and AESModeCtrl.

Address map (AddrMap)

Functionalities of the AddrMap block are:

- Synchronize the AXI AR channel with R channel
- Remaps the address field of the BEE AXI slave interface to address field of the BEE AXI master interface by biasing the received address with a fixed OFFSET
- BEE is capable of remapping two individual regions (Region0 and Region1). Those two regions can be biased with a signed offset (OFFSET0 and OFFSET1). Address field of BEE AXI output (addr_o_0 and addr_o_1) is remapped as below ² :
 - $\text{addr_o_0} = (\text{addr_i_0} + \text{addr_offset0} \ll 16) \% (2^{32})$
 - $\text{addr_o_1} = (\text{addr_i_1} + \text{addr_offset1} \ll 16) \% (2^{32})$

The following diagram shows an example of the address remapping function.

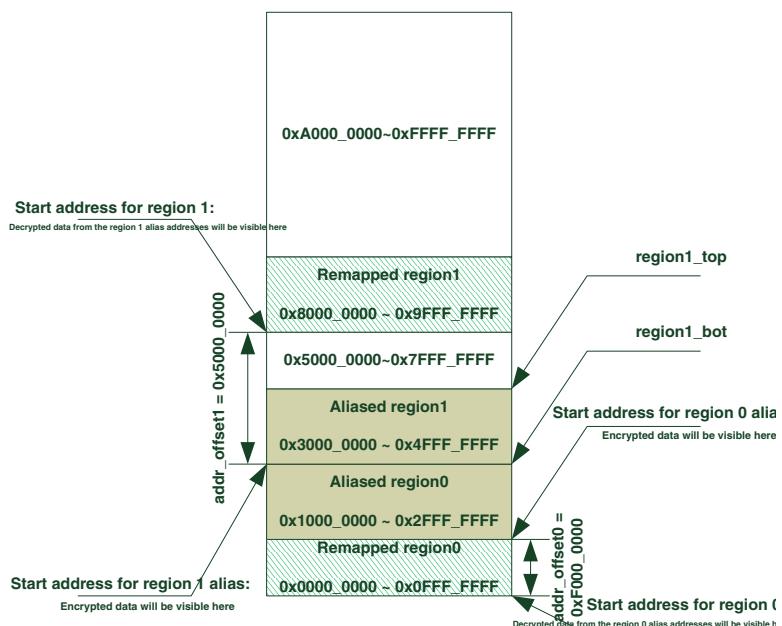


Figure 6-3. Address Space Remap

- Those two aliased space (Aliased region0 and Aliased region1) are [0x1000_0000~0x2FFF_FFFF] and [0x3000_0000~0x4FFF_FFFF]. The memory region is defined by the “region1_top” and “region1_bot” registers. Memory space other than region1 is defined as memory region0.

2. Description of addr_offset0 and addr_offset1 can be found in the BEE Internal Registers section.

- The BEE receives the aliased address and remaps it to a physical address of the encrypted data based on the above requirements.
-

NOTE

When OFFSET0 and/or OFFSET1 are used, applications must ensure that remapped addresses do not point back to the aliased memory regions.

AES algorithmic Core (AESCore)

The AESCore block implements a 128-bit key/data encryption/decryption block as defined by the National Institute of Standards and Technology (NIST) as US FIPS PUB 197 dated November 2001 (see references for specifications and toolkits). There are three variations of AES, each corresponding to the key size used: AES-128, AES-192 or AES-256. AES always operates on 128 bits of data at a time. Only the AES-128 algorithm is implemented at this time.

The core contains separate encrypt and decrypt cipher modules which are instantiated within the AES sub-module. The top-level AES block contains controls to allocate resources between encrypt and decrypt operations and contains a common key expansion module.

Local Buffer 64Byte Control (LBC64B)

To reduce logic gate count, the AESCore applies folded-pipeline architecture which reduces overall throughput of the AESCore. Without proper handling BEE may send non-continuous burst transactions to on-chip fabric which may reduce the bus efficiency. The LBC64B block is added to resolve this problem.

On the read channel the LBC64B:

1. Buffering the incoming read data into local buffer
2. As long as the local buffer isn't empty, it pushes data into the AESCore pipeline
3. After the entire Burst is processed by the AESCore, returns the entire Burst to upstream module in a continuous way

AES mode control (AESModeCtrl)

The BEE module supports both AES-ECB and AES-CTR modes. These two modes are supported by allocating internal resources of the Decryption Engine (DE) by the AESModeCtrl block.

- When BEE works in ECB mode,
 - a. On the decryption path, the Cipher Text (CT) is decrypted by the DE, and the Plain Text (PT) is then returned to upstream module directly.
- When in CTR mode,

BEE internal registers

- ctr_nonce0[127:32] and ctr_nonce1[127:32] as described in the table BEE Internal Register is used as 96-bit CTR modeNonce[127:32] input of region0 and region1
- bit4 to bit31 of the physical address (before BEE address remapping) of corresponding data is used as Counter input
- ctr_nonce0[31:0] and ctr_nonce1[31:0] which corresponds to Nonce[31:0] of region0 and region1 as shown in diagram below is used to control the one-to-one mapping function ($F()$) which maps the address input, Addr[31:4] as shown below, to the final Counter Field (CF). {Nonce[127:32], CF} is then used as the Nonce input for corresponding 16B data block

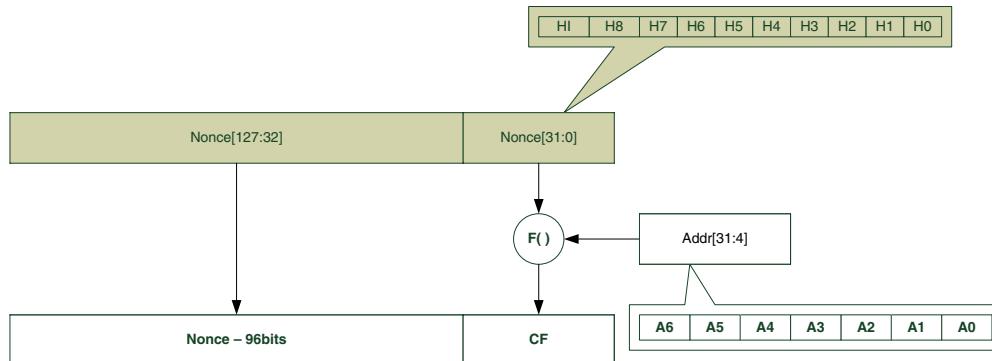


Figure 6-4. CRTNonce Generation

- Functionalities of the $F()$ function is shown as in figure below:

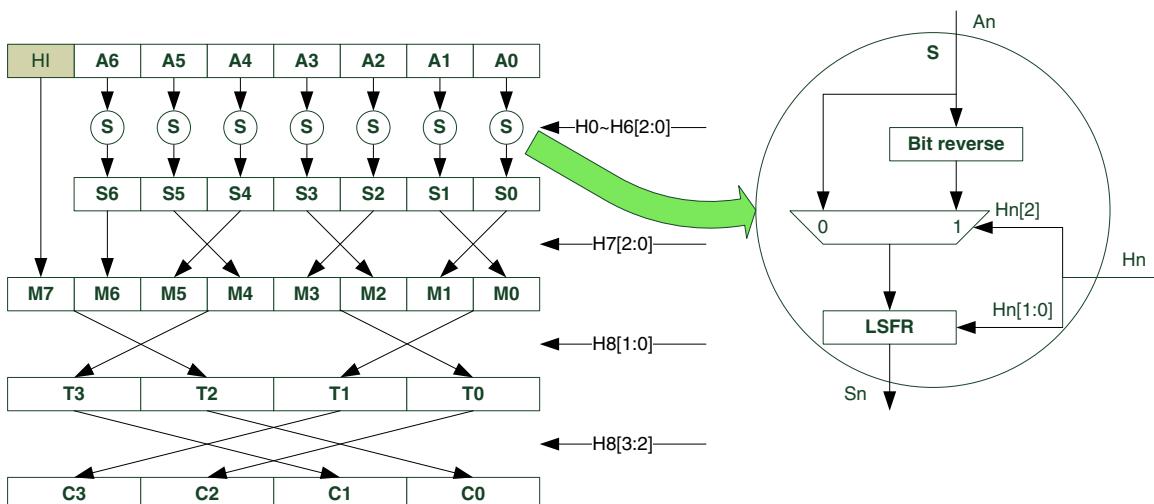


Figure 6-5. Counter Mapping Function

- For decryption, the final Nonce is sent to AESCore for encryption. The AESCore output is then XORed with the CT to generate the final PT output.

6.3 BEE internal registers

BEE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
403E_C000	BEE Control Register (BEE_CTRL)	32	R/W	0000_7700h	6.3.1/200
403E_C004	BEE_ADDR_OFFSET0	32	R/W	0000_F000h	6.3.2/202
403E_C008	BEE_ADDR_OFFSET1	32	R/W	0000_F000h	6.3.3/203
403E_C00C	BEE_AES_KEY0_W0	32	W	0000_0000h	6.3.4/203
403E_C010	BEE_AES_KEY0_W1	32	W	0000_0000h	6.3.5/203
403E_C014	BEE_AES_KEY0_W2	32	W	0000_0000h	6.3.6/204
403E_C018	BEE_AES_KEY0_W3	32	W	0000_0000h	6.3.7/204
403E_C01C	BEE_STATUS	32	R	Undefined	6.3.8/205
403E_C020	BEE_CTR_NONCE0_W0	32	W	0000_0000h	6.3.9/205
403E_C024	BEE_CTR_NONCE0_W1	32	W	0000_0000h	6.3.10/206
403E_C028	BEE_CTR_NONCE0_W2	32	W	0000_0000h	6.3.11/206
403E_C02C	BEE_CTR_NONCE0_W3	32	W	0000_0000h	6.3.12/206
403E_C030	BEE_CTR_NONCE1_W0	32	W	0000_0000h	6.3.13/207
403E_C034	BEE_CTR_NONCE1_W1	32	W	0000_0000h	6.3.14/207
403E_C038	BEE_CTR_NONCE1_W2	32	W	0000_0000h	6.3.15/207
403E_C03C	BEE_CTR_NONCE1_W3	32	W	0000_0000h	6.3.16/208
403E_C040	BEE_REGION1_TOP	32	R/W	0000_0000h	6.3.17/208
403E_C044	BEE_REGION1_BOT	32	R/W	0000_0000h	6.3.18/208

6.3.1 BEE Control Register (BEE_CTRL)

Address: 403E_C000h base + 0h offset = 403E_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REGION1_KEY_LOCK	CTRL_AES_MODE_R1_LOCK	SECURITY_LEVEL_R1_LOCK		REGION0_KEY_LOCK	CTRL_AES_MODE_R0_LOCK	SECURITY_LEVEL_R0_LOCK		LITTLE_ENDIAN_LOCK	AC_PROT_EN_LOCK	KEY_REGION_SEL_LOCK	KEY_VALID_LOCK	REGION1_ADDR_LOCK	CTRL_SFTRST_N_LOCK	CTRL_CLK_EN_LOCK	BEE_ENABLE_LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	CTRL_AES_MODE_R1	SECURITY_LEVEL_R1	Reserved	CTRL_AES_MODE_R0	SECURITY_LEVEL_R0		LITTLE_ENDIAN	AC_PROT_EN	KEY_REGION_SEL	KEY_VALID	Reserved	CTRL_SFTRST_N	CTRL_CLK_EN	BEE_ENABLE	
W																
Reset	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0

BEE_CTRL field descriptions

Field	Description
31 REGION1_KEY_LOCK	Lock bit for region1 AES key
30 CTRL_AES_MODE_R1_LOCK	Lock bit for region1 ctrl_aes_mode
29–28 SECURITY_LEVEL_R1_LOCK	Lock bits for security_level_r1
27 REGION0_KEY_LOCK	Lock bit for region0 AES key
26 CTRL_AES_MODE_R0_LOCK	Lock bit for region0 ctrl_aes_mode

Table continues on the next page...

BEE_CTRL field descriptions (continued)

Field	Description
25–24 SECURITY_LEVEL_R0_LOCK	Lock bits for security_level_r0
23 LITTLE_ENDIAN_LOCK	Lock bit for little_endian
22 AC_PROT_EN_LOCK	Lock bit for ac_prot
21 KEY_REGION_SEL_LOCK	Lock bit for key_region_sel
20 KEY_VALID_LOCK	Lock bit for key_valid
19 REGION1_ADDR_LOCK	Lock bit for region1 address boundary
18 CTRL_SFTRST_N_LOCK	Lock bit for ctrl_sftrst
17 CTRL_CLK_EN_LOCK	Lock bit for ctrl_clk_en
16 BEE_ENABLE_LOCK	Lock bit for bee_enable
15 Reserved	This field is reserved.
14 CTRL_AES_MODE_R1	AES mode of region1 0 ECB 1 CTR
13–12 SECURITY_LEVEL_R1	Security level of the allowed access for memory region1
11 Reserved	This field is reserved.
10 CTRL_AES_MODE_R0	AES mode of region0 0 ECB 1 CTR
9–8 SECURITY_LEVEL_R0	Security level of the allowed access for memory region0
7 LITTLE_ENDIAN	Endian swap control for the 16 bytes input and output data of AES core. 0 The input and output data of the AES core is swapped as below:

Table continues on the next page...

BEE_CTRL field descriptions (continued)

Field	Description
	{B15,B14,B13,B12,B11,B10,B9,B8, B7,B6,B5,B4,B3,B2,B1,B0} swap to {B0,B1,B2,B3,B4,B5,B6,B7, B8,B9,B10,B11,B12,B13,B14,B15}, where B0~B15 refers to Byte0 to Byte15. 1 The input and output data of AES core is not swapped.
6 AC_PROT_EN	Enable access permission control NOTE: When AC_PROT_EN is asserted, all encrypted regions are limited to be ARM core access only.
5 KEY_REGION_SEL	AES key region select 0 Load AES key for region0 1 Load AES key for region1
4 KEY_VALID	AES-128 key is ready
3 Reserved	This field is reserved.
2 CTRL_SFTRST_N	Soft reset input, low active
1 CTRL_CLK_EN	Clock enable input, low inactive
0 BEE_ENABLE	BEE enable bit 0 Disable BEE 1 Enable BEE

6.3.2 BEE_ADDR_OFFSET0

Address: 403E_C000h base + 4h offset = 403E_C004h

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0

BEE_ADDR_OFFSET0 field descriptions

Field	Description
31–16 ADDR_OFFSET0_LOCK	Lock bits for addr_offset0
ADDR_OFFSET0	Signed offset for BEE region 0. The signed offset specified here is added to the upper 16-bits of the region 0 addresses to create aliased regions. The data stored in the region 0 alias is decrypted and visible at the region 0 addresses.

6.3.3 BEE_ADDR_OFFSET1

Address: 403E_C000h base + 8h offset = 403E_C008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR_OFFSET1_LOCK																ADDR_OFFSET1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	

BEE_ADDR_OFFSET1 field descriptions

Field	Description
31–16 ADDR_OFFSET1_LOCK	Lock bits for addr_offset1
ADDR_OFFSET1	Signed offset for BEE region 1. The signed offset specified here is added to the upper 16-bits of the region 1 addresses to create aliased regions. The data stored in the region 1 alias is decrypted and visible at the region 1 addresses.

6.3.4 BEE_AES_KEY0_W0

Address: 403E_C000h base + Ch offset = 403E_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KEY0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

BEE_AES_KEY0_W0 field descriptions

Field	Description
KEY0	AES 128 key from software AES_128KEY={key3,key2, key1,key0}

6.3.5 BEE_AES_KEY0_W1

Address: 403E_C000h base + 10h offset = 403E_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KEY1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

BEE_AES_KEY0_W1 field descriptions

Field	Description
KEY1	AES 128 key from software

BEE_AES_KEY0_W1 field descriptions (continued)

Field	Description
	AES_128KEY={key3,key2, key1,key0}

6.3.6 BEE_AES_KEY0_W2

Address: 403E_C000h base + 14h offset = 403E_C014h

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

Reset 0

BEE_AES_KEY0_W2 field descriptions

Field	Description
KEY2	AES 128 key from software
	AES_128KEY={key3,key2, key1,key0}

6.3.7 BEE_AES_KEY0_W3

Address: 403E_C000h base + 18h offset = 403E_C018h

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

Reset 0

BEE_AES_KEY0_W3 field descriptions

Field	Description
KEY3	AES 128 key from software
	AES_128KEY={key3,key2, key1,key0}

6.3.8 BEE_STATUS

Address: 403E_C000h base + 1Ch offset = 403E_C01Ch

Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	Reserved																
W																	
Reset	x*	x*	x*	x*	x*	x*	x*	x*		x*	x*	x*	x*	x*	x*	x*	x*
Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	Reserved								BEE_IDLE	IRQ_VEC							
W										w1c							
Reset	x*	x*	x*	x*	x*	x*	x*	x*		x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

BEE_STATUS field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 BEE_IDLE	1'b1: BEE is idle; 1'b0: BEE is active
IRQ_VEC	<ul style="list-style-type: none"> • bit 7: Protected region-3 access violation • bit 6: Protected region-2 access violation • bit 5: Protected region-1 access violation • bit 4: Protected region-0 access violation • bit 3: Region-1 read channel security violation • bit 2: Read channel illegal access detected • bit 1: Region-0 read channel security violation • bit 0: Disable abort. Received AXI command during BEE disabled period

6.3.9 BEE_CTR_NONCE0_W0

Address: 403E_C000h base + 20h offset = 403E_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W	NONCE00																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

BEE_CTR_NONCE0_W0 field descriptions

Field	Description
NONCE00	Nonce0 from software for CTR, for region0. Nonce0={Nonce03,Nonce02,Nonce01,Nonce00}

6.3.10 BEE_CTR_NONCE0_W1

Address: 403E_C000h base + 24h offset = 403E_C024h

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

BEE_CTR_NONCE0_W1 field descriptions

Field	Description
NONCE01	Nonce0 from software for CTR, for region0. Nonce0={Nonce03,Nonce02,Nonce01,Nonce00}

6.3.11 BEE_CTR_NONCE0_W2

Address: 403E_C000h base + 28h offset = 403E_C028h

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

BEE_CTR_NONCE0_W2 field descriptions

Field	Description
NONCE02	Nonce0 from software for CTR, for region0. Nonce0={Nonce03,Nonce02,Nonce01,Nonce00}

6.3.12 BEE_CTR_NONCE0_W3

Address: 403E_C000h base + 2Ch offset = 403E_C02Ch

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

BEE_CTR_NONCE0_W3 field descriptions

Field	Description
NONCE03	Nonce0 from software for CTR, for region0. Nonce0={Nonce03,Nonce02,Nonce01,Nonce00}

6.3.13 BEE_CTR_NONCE1_W0

Address: 403E_C000h base + 30h offset = 403E_C030h

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

BEE_CTR_NONCE1_W0 field descriptions

Field	Description
NONCE10	Nonce1 from software for CTR, for region1. Nonce1={Nonce13,Nonce12,Nonce11,Nonce10}

6.3.14 BEE_CTR_NONCE1_W1

Address: 403E_C000h base + 34h offset = 403E_C034h

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

BEE_CTR_NONCE1_W1 field descriptions

Field	Description
NONCE11	Nonce1 from software for CTR, for region1. Nonce1={Nonce13,Nonce12,Nonce11,Nonce10}

6.3.15 BEE_CTR_NONCE1_W2

Address: 403E_C000h base + 38h offset = 403E_C038h

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

BEE_CTR_NONCE1_W2 field descriptions

Field	Description
NONCE12	Nonce1 from software for CTR, for region1. Nonce1={Nonce13,Nonce12,Nonce11,Nonce10}

6.3.16 BEE_CTR_NONCE1_W3

Address: 403E_C000h base + 3Ch offset = 403E_C03Ch

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

BEE_CTR_NONCE1_W3 field descriptions

Field	Description
NONCE13	Nonce1 from software for CTR, for region1. Nonce1={Nonce13,Nonce12,Nonce11,Nonce10}

6.3.17 BEE_REGION1_TOP

Address: 403E_C000h base + 40h offset = 403E_C040h

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

BEE_REGION1_TOP field descriptions

Field	Description
REGION1_TOP	Address upper limit of region1 The REGION1_TOP and REGION1_BOT registers define which FAC addresses (configured in the IOMUXC_GPR registers) will be treated at BEE region 1. Any FAC addresses that do not hit in BEE region1 as defined by REGION1_TOP and REGION1_BOT will be treated as BEE region 0.

6.3.18 BEE_REGION1_BOT

Address: 403E_C000h base + 44h offset = 403E_C044h

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

BEE_REGION1_BOT field descriptions

Field	Description
REGION1_BOT	<p>Address lower limit of region1</p> <p>The REGION1_TOP and REGION1_BOT registers define which FAC addresses (configured in the IOMUXC_GPR registers) will be treated at BEE region 1. Any FAC addresses that do not hit in BEE region1 as defined by REGION1_TOP and REGION1_BOT will be treated as BEE region 0.</p>

6.4 BEE program model

The BEE module is designed to minimize software intervention; however, software assistance is still required in the following scenarios:

- BEE initialization after POR reset
- BEE parameter update, if it is required by the application
- BEE interrupt service routine (ISR)

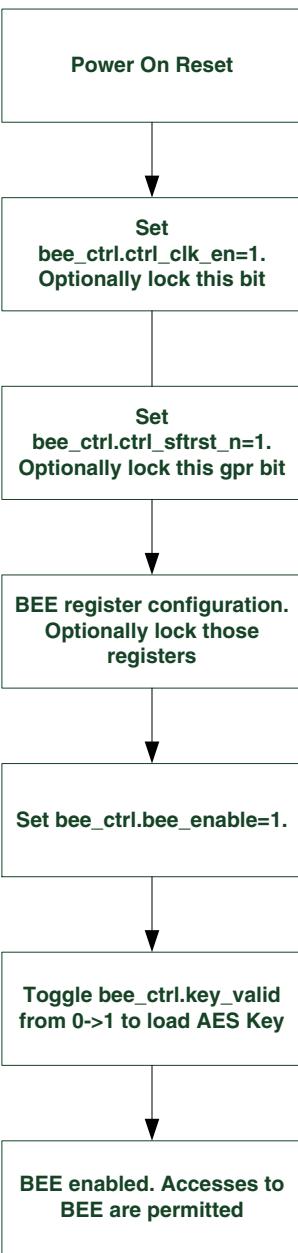


Figure 6-6. BEE initialization flow

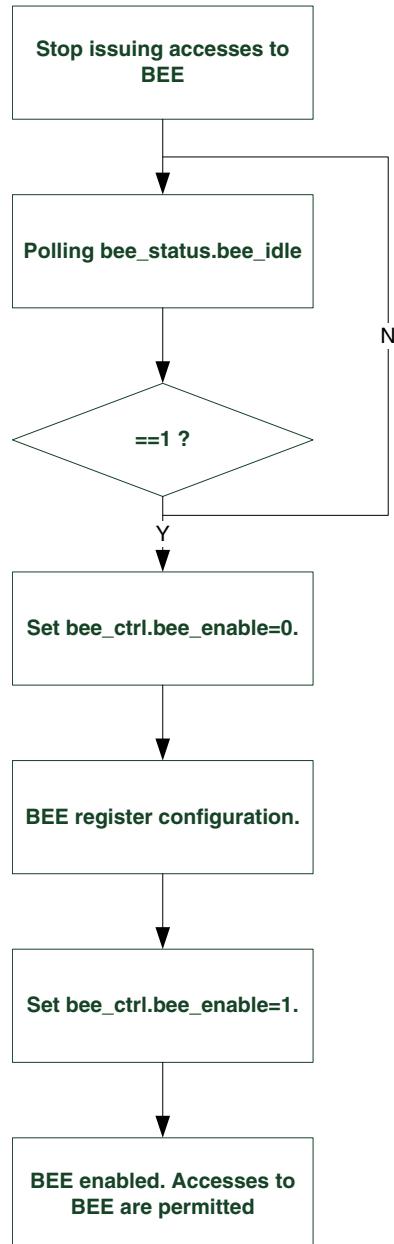


Figure 6-7. BEE parameter update flow

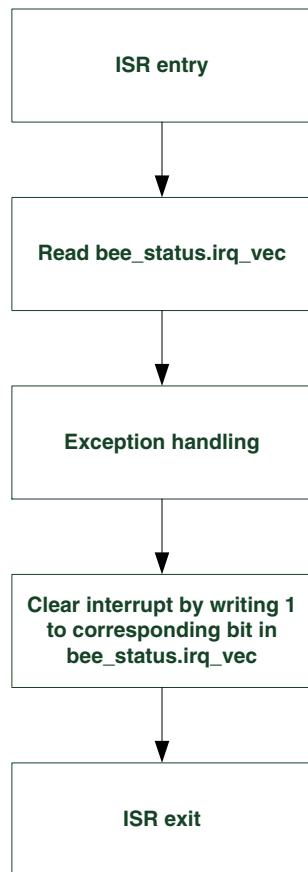


Figure 6-8. BEE ISR flow

Chapter 7

Data Co-Processor (DCP)

7.1 Overview

This chapter describes the Data Co-Processor (DCP) block included on the and how to use it.

It includes sections on memory copy functionality, Advanced Encryption Standard (AES), hashing, and arbitration. The sections on programming the channel operations and example code are also provided. The programmable registers are described in [DCP](#).

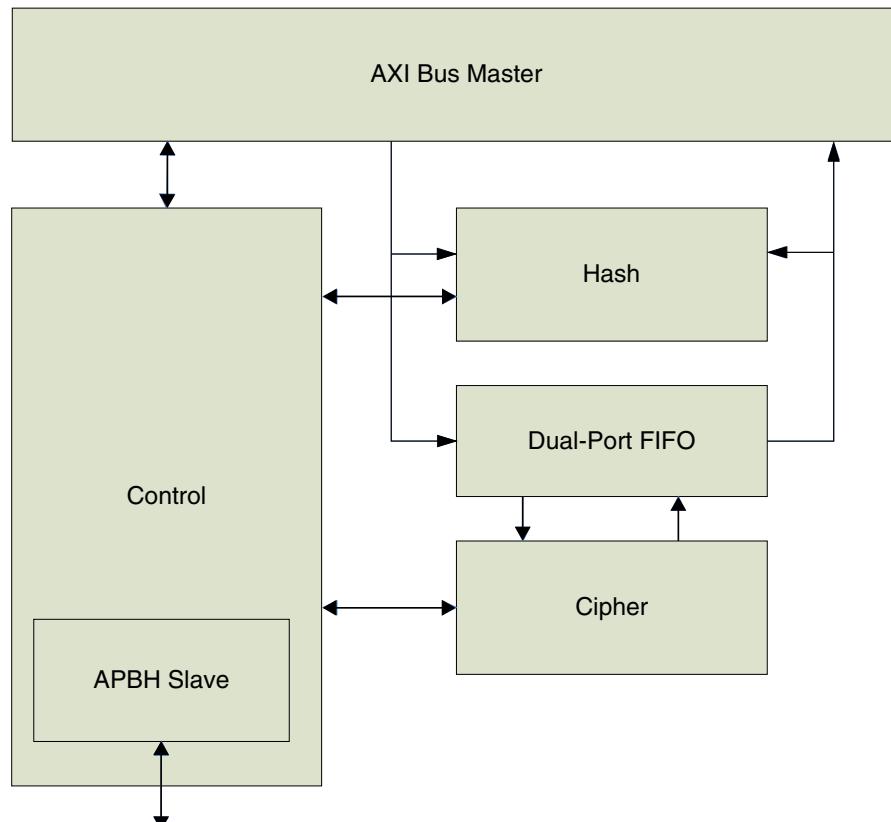


Figure 7-1. DCP block diagram

The DCP module provides support for the general encryption and hashing functions typically used for the security functions. Because its basic job is to move data from memory to memory, it also incorporates the memory-copy (memcpy) function for debugging and as a more efficient method to copy data between memory blocks than the DMA-based approach. The memcpy function also has a "blit" mode of operation where it can transfer a rectangular block of data to a video frame buffer.

The DCP supports a wide variety of encryption and hashing algorithms, and the control structures are designed to enable flexibility in adding of additional algorithms and modes in the future. It supports up to 16 encryption algorithms with up to 16 different modes of operation as well as up to 16 hashing algorithms . While the DCP module is designed to support numerous algorithms, only a subset may be implemented in any given implementation (see the Capability Register section).

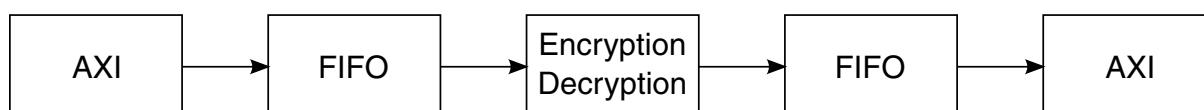
The DCP module processes data based on chained command structures written to the system memory by software (in a manner similar to the DMA engine). The control block maintains the registers to support four independent and concurrent chains, enabling the software to virtualize the access to the DCP block. Each command in the chain represents a work unit that the module processes to completion. At the end of each work unit, the control logic arbitrates among chains with outstanding commands and processes a command from that chain. The arbitration among the channels is round-robin, enabling equal access to the data engine for all active channels. Each channel also supports a "high-priority" mode that enables it to have priority over the remaining channels. If multiple channels are selected as high-priority, the channel arbiter selects from the high-priority channels in a round-robin fashion.

The data flow through the DCP module can be configured in one of five fashions, depending on the functionality activated by the control packet:

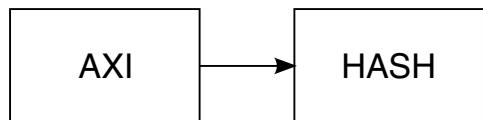
- **Memcpy/blit mode**-The data is moved unchanged from one memory buffer to another.



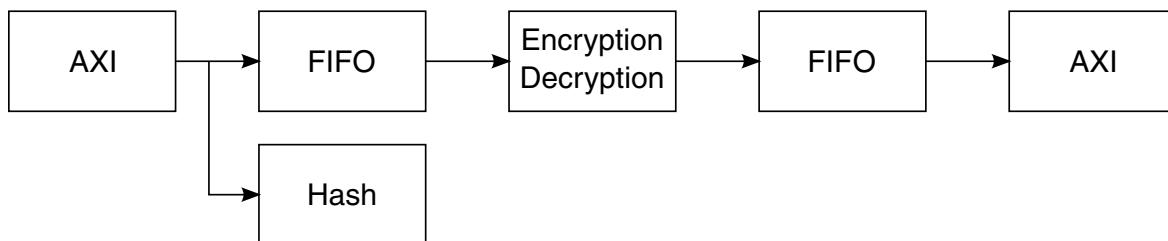
- **Encryption only**-The data from the source buffer is encrypted/decrypted into the destination buffer.



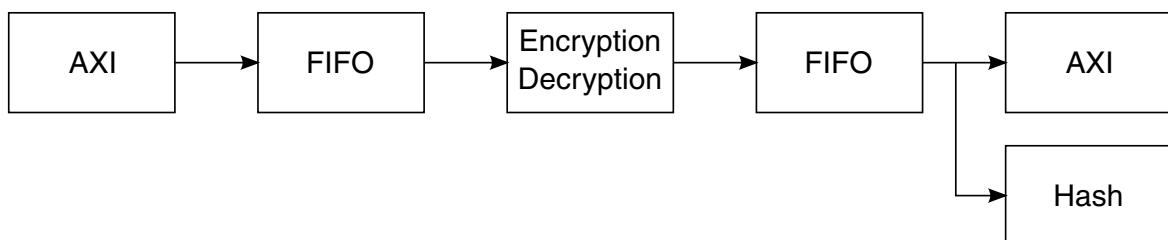
- **Hashing only**-The data from the source buffer is read, and a hash is generated.



- **Encryption and input hashing**-The data from the source buffer is encrypted/decrypted into the destination and the source buffer is hashed.



- **Encryption and output hashing**-The data from the source buffer is encrypted/decrypted into the destination and the output data is hashed.



7.1.1 DCP limitations for software

While the DCP module is designed to be as flexible as possible, there are these limitations to which the software must adhere:

- The buffer sizes for all operations must be aligned to the natural size of the transfer algorithm used. The memcpy operations can transfer any number of bytes (one-byte granularity), and the AES operations must be multiples of 16 bytes (four-word granularity). For all operations, if the byte count is not a word granularity, the hardware rounds up to the next word. The hashing is supported at a byte granularity.
- The DCP module supports buffer operations to any byte alignment, but the performance is improved if the buffers are aligned to a four-byte boundary because the fetch/store operations can be performed without having to make byte operations to accommodate the misaligned addresses.

- The hash operations are limited to a buffer size of 512 MB. The hardware only implements a 32-bit hash length counter instead of the 64-bit counter supported by the SHA-1/SHA-256 algorithm (counter counts bits not bytes, therefore the total of 512 MB).
- For the chained hashing operations (operations involving multiple descriptors), every descriptor except the last must have a byte count that is a 16-word multiple (granularity of the hash algorithm).
- The key values can't be written while the AES block is active. This limitation exists because the key RAM is in use while the AES is operational. Any writes from the peripheral bus cannot be held in the wait states. Therefore, the RAM must be accessible during the key writes.
- The byte-swap controls can only be used with the modulo-4 length buffers. For the non-modulo-4 lengths, the final partial word contains incorrect data. Any address alignment can be used with the byte swapping.
- The word-swap controls are only useful for the cipher operations because the logic forms the 128-bit cipher data from four words from the system memory. The word-swap controls are ignored for the memcpy or hashing operations.
- The DCP only supports writes to the word boundaries to the OCRAM.

7.2 Clocks

See [Clock Controller Module \(CCM\)](#) for the clock setting, configuration, and gating information.

7.3 Operation

The top-level DCP module contains the AXI master, the peripheral slave bus interface units, the main control block and FIFO, and any encryption or hashing blocks included within the design.

The controller manages the fetching of work blocks, the fetching/storing of context information when switching between the chain pointers, and the managing of the data flow through the FIFO, SHA, and AES blocks. The data entering the block from the AXI master is placed in the FIFO for consumption by the cipher block. After the cipher module has finished its operation, the data is placed back into the FIFO and stored back to the memory via the AXI master. When the hashing is enabled, the SHA block takes its inputs from the bus side of the FIFO to enable it to operate without waiting for the cipher block to complete. The peripheral slave provides all register controls and interfaces mainly with the control block.

7.3.1 Memory copy, blit, and fill functionality

In its most basic operation, the DCP supports moving of unmodified data from one place in the system memory to another.

This functionality is referred to as "memcpy", because it operates only on the memory and it copies the data from one place to another. The typical uses for the memcpy might be for fast virtual memory page moves or repositioning of data blocks in the memory. The memcpy buffers can be aligned to any memory address and can be of any length (byte granularity). For the best performance, the buffers must be word-aligned, although the DCP includes the enhancements to improve the performance for the unaligned cases.

The DCP also has the ability to perform basic blit operations that are typical in graphics operations. To specify a blit, the control packet must have the ENABLE_BLIT bit set in the packet control register. The blit source buffers must be contiguous. The output destination buffer for the blit operation is defined as a "M runs of N bytes" that defines a rectangular region in a frame buffer. For the blit operations, each line of the blit can consist of any number of bytes. After performing a run, the DCP updates the destination pointer such that the next destination address falls on the pixel below the start of the previous run operation. This is done by incrementing the starting pointer by the frame buffer width, which is specified in the Control field.

In addition to copying the data within the memory, the DCP also provides a fill operation, where the source data does not come from another memory location, but from an internal register (the source buffer address in the control packet). This is done whenever the DCP_Control0[CONSTANT_FILL] flag is set in the packet control register (see [Control1 field](#)). This feature can be used with the memcpy to prefill the memory with a specified value, or during a blit operation to fill a rectangular region with a constant color.

7.3.2 Advanced Encryption Standard (AES)

The AES block implements a 128-bit key/data encryption/decryption block, as defined by the National Institute of Standards and Technology (NIST) as the US FIPS PUB 197, dated November 2001 (see the references for the specifications and toolkits).

7.3.2.1 Key storage

The DCP implements four SRAM-based keys that can be used by the software to securely store keys on a semi-permanent basis. The keys may be written via the programming interface by specifying a key index (to specify which key to load) and a subword pointer that indicates which word to write within the key. After a subword is written, the logic automatically increments the subword pointer so that the software can program the higher-order words of the key without rewriting the key index. The keys written into the key storage are not readable.

To use a key in the key storage, the cipher descriptor packet must select the key by setting the DCP_Control1[KEY_SELECT] field in the Control1 descriptor field without setting the OTP_KEY or PAYLOAD_KEY fields in the Control0 register. See [Control0 field](#) and [Control1 field](#).

7.3.2.2 AES OTP key

After a system reset, the OTP controller reads the e-fuse devices and provides the OTP key information to the DCP. The DCP receives a 64-bit UNIQUE KEY and a 128-bit CRYPTO KEY. Depending on the key path control fuse, the DCP receives the CRYTPO KEY either directly or indirectly through the SNVS trust controller module. The CRYPTO KEY in fuses is actually 256-bit and a mux (configured by **DCP_KEY_SEL** bitfield in IOMUXC_GPR register set) is used to select the high or low 128 bits of the key. The DCP internally generates the control logic signals to capture this key into the key RAM. The system reset controller coordinates the deassertion of reset such that the OTP key is stable before the DCP reads it. If the SNVS key path is chosen, the security violation removes the CRYPTO KEY, resets the key in the DCP, and makes the key unavailable to the DCP until the next successful secure boot.

To use the OTP key, the descriptor packet must set the OTP_KEY field in the Control1 register (see [Control1 field](#)).

7.3.2.3 Encryption modes

The most basic form of encryption is the Electronic Code Book (ECB) mode. In this mode, the encryption output is a function of the key and the plaintext only, thus it can be visualized as a giant look-up table. While this provides a great deal of security, there are some limitations. For instance, if the same plaintext appears in a block of data more than once, the same ciphertext appears also. This can be evident if the plaintext contains large blocks of constant data (for example zeros) and can be used to formulate the attacks against a key.

To make the ciphers stronger, several modes of operation can be implemented around the basic ECB cipher to provide additional security. One such mode is the CBC mode (Cipher Block Chaining) which takes the previous encrypted data and logically XORs it with the next incoming plaintext before performing the encryption. During the decryption, the process is reversed and the previously encrypted data is XORed with the decrypted ECB data to provide the plaintext again.

The AES engine supports handling of various modes of operation. The core AES block supports the ECB mode, and the other algorithms are handled in the wrapper around the encryption blocks. The DCP module supports Cipher Block Chaining (CBC), which chains the data blocks by XORing the previously encrypted data with the plaintext before the encryption. The CBC encryption is shown in this figure:

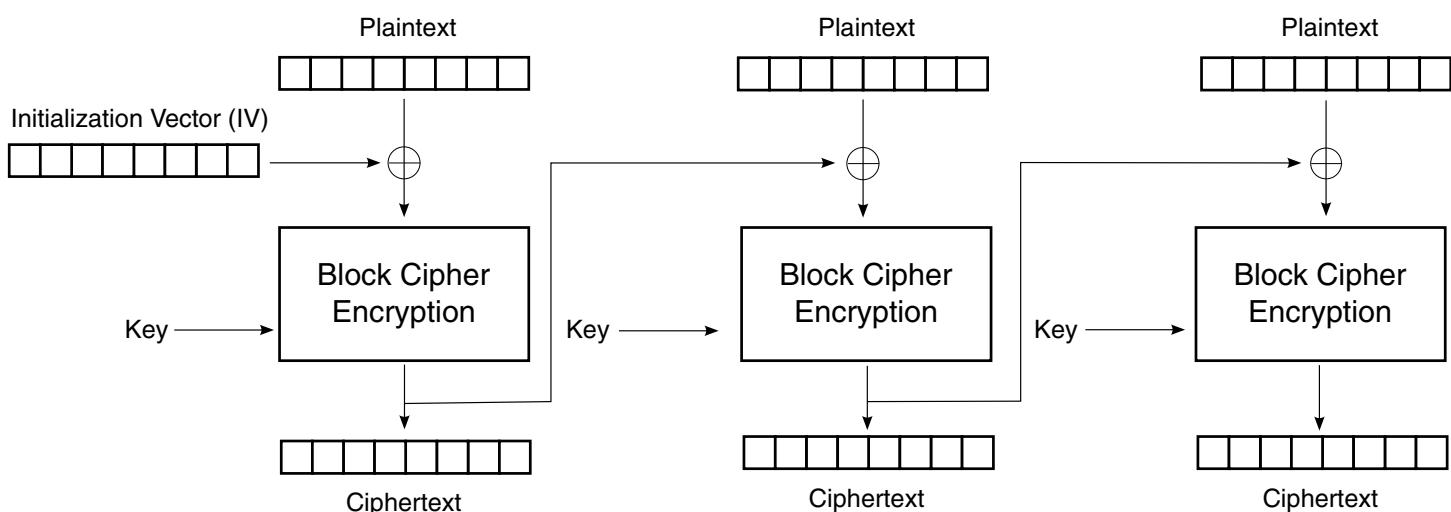


Figure 7-2. CBC mode encryption

The decryption (shown in the following figure) works in a similar manner, where the cipher text is first decrypted and then XORed with the previous ciphertext. For the first encryption/decryption operation, the Initialization Vector (IV) is used to seed the operation. The IV must be the same for both the encryption and decryption steps. Otherwise, the decrypted data don't match the original plaintext.

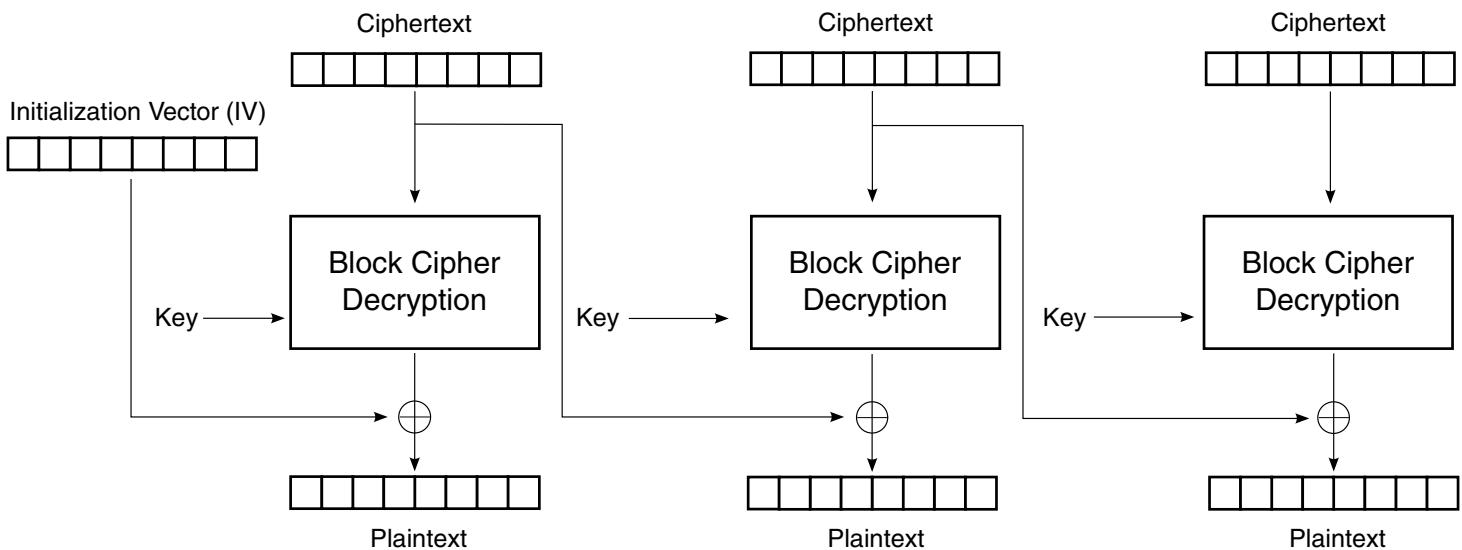


Figure 7-3. CBC mode decryption

7.3.3 Hashing

The hashing module implements the SHA-1 and SHA-256 hashing algorithms and a modified CRC-32 checksum algorithm.

These algorithms produce a signature for a block of data that can be used to determine whether the data is intact.

The CRC-32 algorithm implements a 32-bit CRC algorithm similar to the one used by Ethernet and many other protocols. The CRC differs from the Unix `csum()` function in these three ways:

- The CRC is initialized as 0xFFFFFFFF instead of 0x00000000.
- The logic pads the zeros to a 32-bit boundary for the trailing bytes.
- The logic doesn't post-pend the file length.

The SHA-1 block implements a 160-bit hashing algorithm that operates on 512-bit (64-byte) blocks, as defined by US FIPS PUB 180-1 in 1995. The SHA-256 mode implements a 256-bit hashing algorithm that operates on 512-bit (64-byte) blocks, as defined by US FIPS PUB 180-2 in 2002. The purpose of the hashing module is to generate a unique signature for a block of data that can be used to validate the integrity of the data by comparing the resulting "digest" with the original digest.

The results of the hash operations are written to the beginning of the payload for the descriptor. The DCP also has the ability to check the resulting hash against a value in the payload and issue an interrupt if a mismatch occurs.

7.3.4 One-Time Programmable (OTP) key

After a system reset, the OTP controller reads the e-fuse devices and provides the OTP key information to the DCP through the 128-bit otp_crypto_key_data input data bus and the 64-bit otp_unique_key_data unique key input that come directly from the OTP controller.

These bus input values are held constant and originate from the eFuse shadow registers after a reset. Note that the shadow registers for these bits can be changed in the OTP block if they are unlocked. They are only loaded by the DCP after the deassertion of the system reset. After a reset, the DCP automatically loads these (at the time 32 bits wide) keys into the internal key RAM.

While the OTP key is normally not available for the read operations, the module enables the key to be read if the otp_crypto_key_ren (read-enable) input is active (high). This enables the verification of the key after it is programmed into the eFuse device. After the programming is validated, another fuse is set to disable the read capability.

The OTP key (CRYPTO KEY) can be selected using the DCP_Control1[OTP_KEY] bit in the control field of the packet descriptor or by using the key select 0xFF in the CTRL1 field of the descriptor. The DCP also supports a second hardware key called the UNIQUE KEY which is generated from the OTP KEY and the key modifier bits from other OTP fuse fields. This key is unique to the device and can be used to encrypt the private data stored on the NAND. This key can be selected by writing 0xFE to the KEY_SELECT field in the CTRL1 packet data.

7.3.5 Managing DCP channel arbitration and performance

The DCP can have four channels and all competition for the DCP resources to complete their operations.

Depending on the situation, the critical operations may need to be prioritized above the less important operations to ensure a smooth system operation. To help the software achieve this goal, the DCP implements a programmable arbiter for the internal DCP operations and provides the recovery timers on each channel to throttle the channel activity.

7.3.5.1 DCP arbitration

The DCP implements a multi-tiered arbitration policy that gives the software the flexibility in scheduling the DCP operations. This figure illustrates the arbitration levels and how each channel fits into the arbitration scheme:

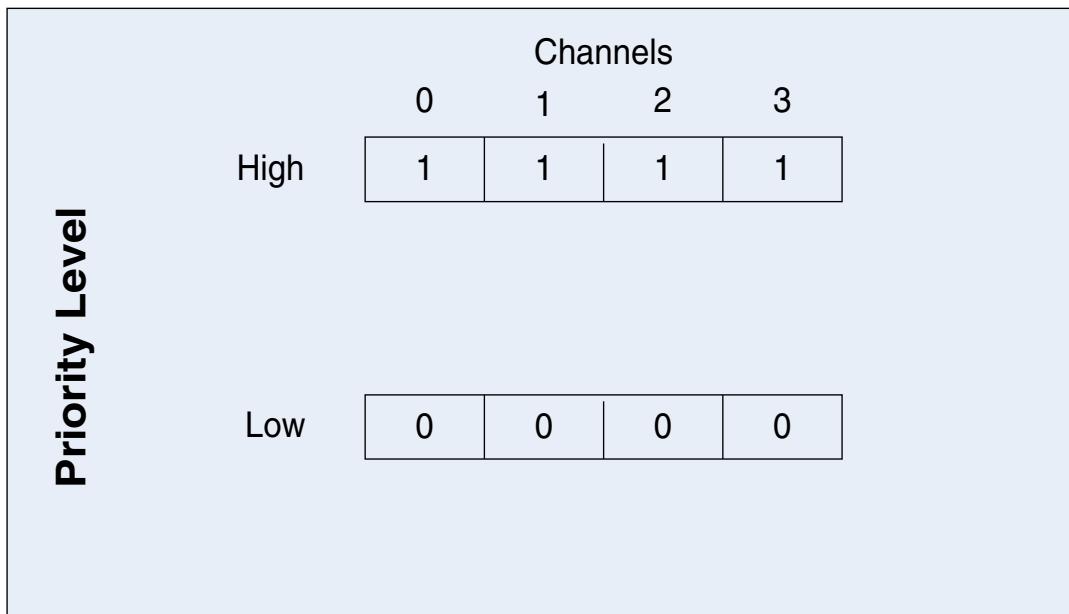


Figure 7-4. DCP arbitration

Each channel can be programmed as being in either the high-priority or the low-priority arbitration pool, depending on the setting in the HIGH_PRIORITY_CHANNEL field of the DCP_CHANNELCTRL register. When the corresponding bit is programmed as a 1, the channel arbitrates in the high-priority pool. Otherwise, it arbitrates in the low-priority pool. When a channel is selected, it completes one packet, and then the arbiter rearbitrates. The channel that just completed participates in the new arbitration round.

7.3.5.2 Channel-recovery timers

Each channel contains a channel-recovery timer in its DCP_CHnOPTS register. The purpose of the recovery timer is to keep the channel inactive for a period of time after it completes an operation. This capability can be used for a high-priority channel to ensure that at least some of the lower-priority requests get serviced between the packets or to simply enable more timeslices for the other channels to perform the operations. The value programmed into the recovery timers register delays the channel from operations until it is 16 times the programmed value. Any non-zero value must prevent the channel from participating in the next arbitration cycle.

7.3.6 Programming channel operations

The control logic block maintains the channel pointers and manages the arbitration and context switching between the different channels.

It also manages the fetching of work packets and the data fetch/store operations from the AXI master interface and coordinates the actions of the hashing and encryption blocks.

The control logic maintains four channels that enable the software to effectively create four independent work sets for the DCP module. The software can construct the chained control packets in the memory that describe the encryption/hashing/memcopy operations to the hardware unit. The address for this first control packet can be written to one of the four virtual channels and enabled. When one or more of the channels is enabled, the controller fetches the control packet pointed to by that channel and initiates the data fetches from the source buffer. The data is then processed as described in the control packet and stored back to the system memory.

When the processing for a control packet is complete, the controller writes the completion status information back to the control packet, and optionally stores the relevant state information to the context buffer. If the control packet specifies a subsequent control packet, the channel's pointer is updated to the address for the next packet and an optional interrupt can be issued to the processor.

At this point, the DCP module arbitrates among the virtual channels for the next operation and processes its control packet. If a subsequent operation continues from a previous operation, the controller automatically loads the context from the previous session into the working registers before resuming the operation for that channel.

7.3.6.1 Virtual channels

The DCP module processes the data via the work packets stored in the memory. Each channel contains a pointer to the current work packet and enough control logic to determine whether the channel is active and to provide the status to the processor. Each channel also provides a recovery timer to help throttle the processing by a particular channel. After processing a packet, the channel enables its 16-bit recovery timer (if the recovery time is set to a non-zero value). The channel does not become active again until its recovery timer expires. The recovery timer timebase is 16 HCLK cycles, so the timer acts as a 20-bit timer with the bottom four bits implicitly tied to 0. This provides an effective range from zero to 2^{20-1} clocks or 0 ns to 7.8 ms at 133 MHz.

A channel is activated any time its semaphore is non-zero and its recovery timer is cleared. The semaphore can be incremented by the software to indicate that the chain pointer is loaded with a valid pointer. As the hardware completes the work packets, it decrements the semaphore if the Decrement Semaphore flag in the Control 0 field is set. The work packets can be chained together using the CHAIN or CHAIN_CONTIGUOUS bits in the Control0 field which causes the channel to automatically update the work packet pointer to the value in the NEXT_COMMAND_ADDRESS field at the end of the current work packet.

7.3.6.2 Context switching

The control logic maintains four virtual channels that enable the DCP block to time-multiplex encryption, hashing, and memcpy operations. It must also retain the state information when changing the channels so that when a channel is resumed, it can resume the operation from where it left off. This process is called context switching.

To minimize the number of registers used in the design, the controller saves the context information from each channel into a private context area in the system memory. When initializing the DCP module, the software must allocate the memory for the context buffer and write the address into the context buffer pointer register. As the DCP module processes the packets, it saves the context information for each channel to the buffer after completing each control packet. When the channel is subsequently activated, the DCP module's internal registers are then reloaded with the proper context before resuming the operation.

Each channel reserves one-fourth of the context buffer area for its context storage. The context buffer consumes 208 bytes of the system memory and is formatted as shown in this figure:

Table 7-1. DCP context buffer layout

Range	Channel	Data	Size
0x00-0x0C	3	Cipher context	16 B
0x10-0x30	-	Hash context	36 B
0x34-0x40	2	Cipher context	16 B
0x44-0x64	-	Hash context	36 B
0x68-0x74	1	Cipher context	16 B
0x78-0x98	-	Hash context	36 B
0x9C-0xA8	0	Cipher context	16 B
0xAC-0xCC	-	Hash context	36 B

The control logic writes to the context buffer only if the function is being used. This means that the cipher context is stored for the CBC encryption/decryption operations only, and the hash context is written only if SHA-1/SHA-256 is used. If neither of these modes are used for a given channel, the memory for the context buffer does not need to be allocated by the software.

Since channel 0 is likely to be used for the VMI in an SDRAM-based system-to-page data from the SDRAM to the on-chip SRAM, the buffer allocation table is organized so that the *highest* numbered channel uses the *lowest* area in the context buffer. For this reason, the software must allocate the higher-numbered channels for the encryption/ hashing operations and the lower-numbered channels for the memcpy operations to reduce the size of the context buffer.

If only a single channel is used for the CBC mode operations or hashing operations, the controller provides a bit in the control register to disable the context switching. In this scenario, the context switching is not required because the other channels do not corrupt the state of the hashing or cipher modes. When the channel resumes, a context load is not required.

Additionally, the control logic monitors the use of CBC/hashing, so that a context reload is not done when the previous channel resumes an operation without an intermediate operation from another channel.

7.3.6.3 Working with semaphores

Each channel has a semaphore register to indicate that the control packets are ready to be processed. Several techniques can be used when programming the semaphores to control the execution of packets within a channel. The channel continues to execute the packets as long as the semaphore is non-zero, a chain bit is set in the descriptor, and no error occurred.

- The software can write the number of pending packets into the semaphore register with the Decrement Semaphore bit in each control packet set. In this scenario, the channel simply decrements the semaphore for each packet set and terminates at the end of the packet chain. The benefit of this method is that the software can easily determine how many packets have executed by reading the semaphore status register.
- The software can create a packet chain with the Decrement Semaphore bit set only in the last packet. In this case, the software writes 1 to the semaphore register to start the chains and the DCP terminates after executing the last packet.
- The software can create a packet chain with the Decrement Semaphore bit set for each packet and write either 1 or a number less than the number of packets to the semaphore register. This can be useful when debugging, because it enables the

channel to execute only a portion of the packets and the software can inspect the intermediate values before restarting the channel again.

If an error occurs, the channel issues an interrupt and clears the semaphore register. The channel does not perform any further operations until the error bits in the status register are cleared. The software can manually clear a non-zero semaphore by writing 0xFF to the CLR (clear) address of the semaphore register.

7.3.6.4 Work packet structure

The work packets for the DCP module are created in the memory by the processor. Each work packet includes all information required for the hardware to process the data. The general structure of the packet is shown in this figure:

Word0	Next Cmd Addr
Word1	Control0
Word2	Control1
Word3	Source Buffer Addr
Word4	Destination Buffer Addr
Word5	Buffer Size
Word6	Payload Pointer
Word7	Status

Figure 7-5. DCP work packet structure

For some operations, additional information is required to process the data in the packet. This additional information is provided in the variable-sized payload buffer which can be found at the address specified in the payload pointer field. When the encryption is specified, the payload may include the encryption key (if the key selected resides in the packet), the initialization vector (for modes of operation such as CBC), and the expected hash values when the data hashing is indicated. The hardware can automatically compare the expected hash with the actual hash and interrupt the processor only on a mismatch. The payload area is used by the software to store the calculated hash value at the end of the hashing operations (when the HASH_TERM control bit in DCP_Control0 is set).

7.3.6.4.1 Next command address field

The NEXT_COMMAND_ADDRESS field (as shown in the following figure) is used to point to the next work packet in the chain. This field is loaded into the channel's command pointer after the current packet has completed processing if the CHAIN bit in the CONTROL0 field (see [Table 7-3](#)) is set. The next command address field must be programmed to a non-zero value when the CHAIN bit is set. Otherwise, the channel flags an invalid setup error.

Table 7-2. DCP next command address field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															

7.3.6.4.2 Control0 field

The main functions of the DCP module are enabled with the ENABLE_MEMCOPY, ENABLE_BLIT, ENABLE_CIPHER, and ENABLE_HASH bits from the Control0 field in the work packet.

The combinations of these bits determine the function performed by the DCP. [Table 7-4](#) summarizes the function performed for each combination.

Table 7-3. DCP Control0 field

TAG	OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWAP	TEST_SEMA IRQ	CONSTANT_FILL	HASH_OUTPUT	HASH_CHECK	HASH_TERM	HASH_INIT	PAYOUT_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYP	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHER	ENABLE_MEMORY	CHAIN_CONTINUOUS	CHAIN	DECR_SEMAPHOR	INTERRUPT_ENABL	E
-----	-----------------	-----------------	----------------	----------------	--------------	--------------	---------------	---------------	-------------	------------	-----------	-----------	------------	---------	-------------	---------------	-------------	-------------	---------------	---------------	------------------	-------	---------------	-----------------	---

Table 7-4. DCP function enable bits

Hash	Cipher	Blit	Memcpy	Description
0	0	0	X	Simple memcpy operation.
0	0	1	0	Blit operation.
0	1	0	0	Simple encrypt/decrypt operation.
1	0	0	0	Simple hash. Only reads performed.
1	0	0	1	Memcpy and hash operation.
1	1	0	0	Hash with encryption/decryption.
All others				Invalid setup.

The CHAIN bit must be set if the NEXT_COMMAND_ADDRESS field has a valid pointer to the next work packet. When set, this bit causes the channel to update its pointer to the next packet. The CHAIN_CONTINUOUS bit is similar, but it indicates that the next packet follows immediately after the current packet. This enables the software to generate templates of operations without regards to the actual physical addresses used, which makes programming easier (especially in a virtual memory environment).

If the INTERRUPT_ENABLE bit is set, the channel generates an interrupt to the processor after completing the work for this packet. When the interrupt is issued, the packet would be completely processed, including the update of the status/payload fields in the work packet.

Each channel contains an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands. Whenever a command finishes its operation, it checks the DECR_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the idle state and remains there until the semaphore is incremented by the software. When the semaphore goes to non-zero and the channel is in its idle state, it then uses the value in the NEXT_COMMAND_ADDRESS field to begin processing.

If the ENABLE_CIPHER bit is set, the data is encrypted or decrypted based on the CIPHER_ENCRYPT bit using the key/encryption settings from the Control1 field. The OTP_KEY and PAYLOAD_KEY indicate the source of the keys for the operation. If the OTP_KEY value is set, the KEY_SELECT field from the Control1 register indicates which OTP key is to be used. If the PAYLOAD_KEY bit is set, the first entry in the payload is the key to be used for the operation. If neither bit is set, the KEY_SELECT field indicates an index in the key RAM that contains the key to be used. (For cases when both the OTP_KEY and PAYLOAD_KEY bits are set, the PAYLOAD_KEY takes precedence.)

The HASH_ENABLE enables hashing the data with the HASH_OUTPUT bit controlling whether the input data (HASH_OUTPUT=0) or the output data (HASH_OUTPUT=1) is hashed. In addition, the hardware can be programmed to automatically check the hashed data if the HASH_CHECK bit is set. If the hash does not match, an interrupt is generated, and the channel terminates the operation on the current chain. The hashing algorithms also require two additional fields to operate properly. The HASH_INIT bit must be set for the first block in a hashing pass to properly initialize the hashing function. The HASH_TERM bit must be set on the final block of a hash to notify the hardware that the hashing operation is complete so that it can properly pad the tail of the buffer according to the hashing algorithm. This flag also triggers the write-back of the hash output to the work packet's payload area.

The CONSTANT_FILL flag may be used for both the memcpy and blot operations. When this is set, the source address field is used as a constant that is written to all locations in the output buffer instead.

The WORDSWAP and BYTESWAP bits enable the muxes around the FIFO to swap the data to handle the little-endian and big-endian storage of data in the system memory. When these bits are cleared, the data is assumed to be in the little-endian format. When all bits are set, the data is assumed to be in the big-endian format.

The TAG field is used to identify the work packet and the associated completion status. When each packet is completed, the channel provides the status and tag information for the last packet processed.

7.3.6.4.3 Control1 field

The Control1 field (shown in the following table) provides additional values used when hashing or encrypting/decrypting data.

For the blot operations, this field indicates the number of bytes in a frame buffer line, which is used to calculate the address for successive lines in each blot operation.

Table 7-5. DCP Control1 field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CIPHER_CONFIG								HASH_SELECT	KEY_SELECT								CIPHER_MODE				CIPHER_SELECT										
FRAMEBUFFER_LENGTH (in bytes, blot mode)																															

The CIPHER_SELECT field selects one from the sixteen possible encryption algorithms (0=AES128). The CIPHER_MODE selects the mode of operation for that cipher (0=ECB, 1=CBC).

The KEY_SELECT field enables the key selection for one of several sources. The keys can be included in the packet payload or may come from the OTP or write-only registers.

The HASH_SELECT field selects a hashing algorithm for the operation (0=SHA-1, 1=CRC32, 2=SHA-256).

The CIPHER_CONFIG field provides the optional configuration information for the selected cipher. An example is a key length for the RC4 algorithm.

7.3.6.4.4 Source buffer

The SOURCE_BUFFER pointer (shown in the following figure) specifies the location of the source buffer in the memory. The buffer can reside at any byte alignment and must refer to the on-chip SRAM or the off-chip SDRAM location. For optimal performance, the buffers must be word-aligned. When the CONSTANT_FILL flag is set in the Control0 field, the value in this field is used as the data written to all destination buffer locations.

Table 7-6. DCP source buffer field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SOURCE_BUFFER																															
CONSTANT (CONSTANT_FILL mode)																															

7.3.6.4.5 Destination buffer

The DESTINATION_BUFFER (shown in the following table) specifies the location of the destination buffer in the on-chip SRAM or the off-chip SDRAM memory and can be set to any byte alignment. For the in-place encryption, the destination buffer and the source buffer must be of the same value. For the optimal performance, the buffer location must be word-aligned.

Table 7-7. DCP destination buffer field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DESTINATION_BUFFER																															

7.3.6.4.6 Buffer size field

The BUFFER_SIZE field (shown in the following table) indicates the size of the buffers for the memcpy, encryption, and hashing modes. For the memcpy and hashing operations, the value can be any number of bytes (byte granularity). For the encryption modes, the length must be a multiple of the selected encryption algorithm's natural data size (16 B for AES).

Table 7-8. DCP buffer size field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NUMBER_LINES (blit mode)																BLIT_WIDTH (in bytes, blit mode)															
BUFFER_SIZE (in bytes, memcpy, encryption, and hashing modes)																															

For blit operations, the buffer size field is split into two portions: a BLIT_WIDTH value which specifies the number of bytes in each line of the blit operation, and a NUMBER_LINES value which specifies how many vertical rows of pixels are in the image buffer.

7.3.6.4.7 Payload pointer

Some operations require additional control values that are stored in the payload buffer (shown in the following table) which is pointed to by this field. After the DCP reads the control packet, it examines the Control0 register and determines whether any payload information is required. If so, the DCP loads the payload from the address specified in this field. (See [Payload](#) for more details.)

Table 7-9. DCP payload buffer pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PAYLOAD_POINTER																															

The payload area is also written to by the DCP at the completion of a hash operation (when the HASH_TERM bit is set). The software must allocate the appropriate amount of storage (20 B for SHA-1 and 4 B for CRC32) in the payload or risk having the DCP write to an unallocated address.

7.3.6.4.8 Status

After the DCP engine has completed processing the descriptor, it writes the packet status (shown in the following table) back to the descriptor in the status field. The packet status is the value of the channel status register at the time the packet completed processing.

Table 7-10. DCP status field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG																															

ERROR_CODE
E

Operation

For various error conditions, the DCP encodes additional error information in the ERROR_CODE field. See [DCP](#) for more details on the assignments of error codes. For any completion codes besides the COMPLETE flag, the channel suspends the processing of the command chain until the error status values are cleared in the channel status register.

The format for this field is the same as for the channel status register, with the exception that the COMPLETE bit is written to the payload status but is not present in the channel status register.

7.3.6.4.9 Payload

The payload is a variable-length field that is used to provide the key, initialization values, and expected results from the hashing operations.

The payload may consist of the data fields listed in this table:

Table 7-11. DCP payload field

Field name	Size	Description	Condition
Cipher key	16 B	AES key	Cipher enable with the PAYLOAD_KEY
Cipher IV	16 B	CBC initialization vector	Cipher enable with the CBC mode
Hash check	20 B	Hash completion value	Hash enabled with the HASH_TERM fields set

If the fields are not used, they do not appear in the payload and the other payload values move upwards in the payload area. For example, if only hashing is used, then the HASH_CHECK value appears at offset 0 in the payload area. This table must be used by the software to determine the amount of payload to allocate and initialize:

Table 7-12. DCP payload allocation by software

Control bits present			Payload size
Hash_Check	Cipher_Init	Payload_Key	
0	0	0	0 words / 0 bytes
0	0	1	4 words / 16 bytes
0	1	0	4 words / 16 bytes
0	1	1	8 words / 32 bytes
1	0	0	SHA-1: 5 words / 20 bytes SHA-2: 8 words / 32 bytes
1	0	1	SHA-1: 9 words / 36 bytes SHA-2: 12 words / 48 bytes
1	1	0	SHA-1: 9 words / 36 bytes SHA-2: 12 words / 48 bytes

Table continues on the next page...

Table 7-12. DCP payload allocation by software (continued)

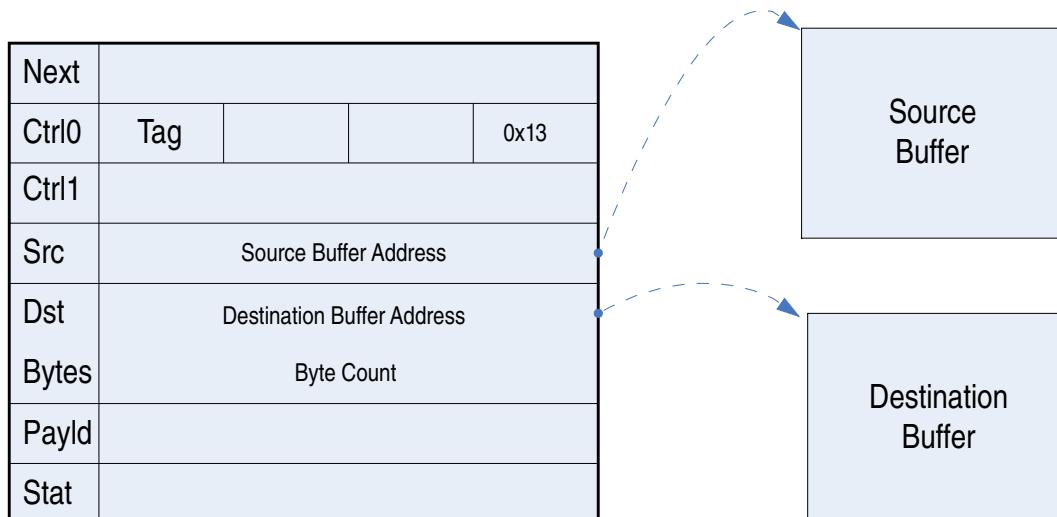
Control bits present			Payload size
Hash_Check	Cipher_Init	Payload_Key	
1	1	1	SHA-1: 13 words / 52 bytes SHA-2: 16 words / 64 bytes

For the hashing operations, the DCP module writes the final hash value back to the start of the payload area for the descriptors with the HASH_TERM bit set in the control packet. It is important that the software allocates the required payload space, even though it is not required to set up the payload for control purposes.

7.3.7 Programming DCP functions

7.3.7.1 Basic memory copy programming example

To perform a basic memcpy operation, only a single descriptor is required. The DCP simply copies the data from the source to the destination buffer. This process is illustrated in this figure:



0x13 = Memcopy | DecrSema | Interrupt

Figure 7-6. Basic memory copy operation

```

<code>
typedef struct _dcp_descriptor
{
    u32          *next;
    hw_dcp_packet1_t ctrl0;
    hw_dcp_packet2_t ctrl1;
    u32          *src,
    *dst,

```

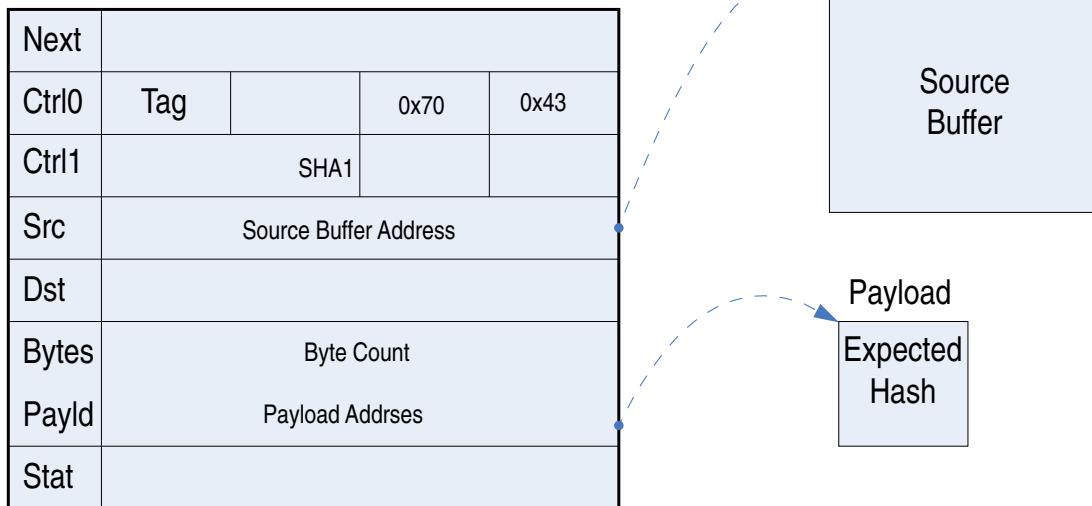
```

        buf_size,
        *payload,
        stat;
} DCP_DESCRIPTOR;
DCP_DESCRIPTOR dcp1;
u32 *srcbuffer, *dstbuffer;
// set up control packet
dcp1.next = 0;                                // single packet in chain
dcp1.ctrl0.U = 0;                             // clear ctrl0 field
dcp1.ctrl0.B.ENABLE_MEMCOPY = 1; // enable memcpy
dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp1.ctrl0.B.INTERRUPT = 1; // interrupt
dcp1.ctrl1.U = 0;                             // clear ctrl1
dcp1.src = srcbuffer;                         // source buffer
dcp1.dst = dstbuffer;                         // destination buffer
dcp1.buf_size = 512;                          // 512 bytes
dcp1.payload = NULL;                          // not required
dcp1.status = 0;                            // clear status
// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1); // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 1); // increment semaphore by 1
// now wait for interrupt or poll
// polling code
while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>

```

7.3.7.2 Basic hash operation programming example

To perform a basic hash operation, only a single descriptor is required. The DCP simply reads the data from the source buffer and computes the hash value on the contents. This process is illustrated in this figure:



0x70 = Hash Check | Hash Term | Hash Init

0x43 = Hash Enable | DecrSema | Interrupt

Figure 7-7. Basic hash operation

```

<code>
typedef struct _dcp_descriptor
{
    u32             *next;
    hw_dcp_packet1_t ctrl0;
    hw_dcp_packet2_t ctrl1;
    u32             *src,
    *dst,
    buf_size,
    *payload,
    stat;
} DCP_DESCRIPTOR;
DCP_DESCRIPTOR dcp1;
u32 *srcbuffer;
u32 payload[5];
// set up expected hash check value
payload[0]=0x01234567;
payload[1]=0x89ABCDEF;
payload[2]=0x00112233;
payload[3]=0x44556677;
payload[4]=0x8899aab;
// set up control packet
dcp1.next = 0;                      // single packet in chain
dcp1.ctrl0.U = 0;                    // clear ctrl0 field
dcp1.ctrl0.B.HASH_CHECK = 1;         // check hash when complete
dcp1.ctrl0.B.HASH_INIT = 1;          // initialize hash with this block
dcp1.ctrl0.B.HASH_TERM = 1;          // terminate hash with this block
dcp1.ctrl0.B.ENABLE_HASH = 1;         // enable hash
dcp1.ctrl0.B.DECR_SEMAPHORE = 1;      // decrement semaphore
dcp1.ctrl0.B.INTERRUPT = 1;          // interrupt
dcp1.ctrl1.U = 0;                   // clear ctrl1
dcp1.src = srcbuffer;                // source buffer
dcp1.dst = 0;                       // no destination buffer
dcp1.buf_size = 512;                 // 512 bytes
dcp1.payload = payload;              // holds expected hash value
dcp1.status = 0;                     // clear status
// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1);      // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 1);            // increment semaphore by 1
// now wait for interrupt or poll
// polling code

```

Operation

```
while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>
```

7.3.7.3 Basic cipher operation programming example

To perform a basic cipher operation, only a single descriptor is required. The DCP reads the data from the source buffer, encrypts it, and writes it to the destination buffer. For this example, the key is provided in the payload and the algorithm uses the AES CBC mode of operation. This requires a payload with both the key and the CBC initialization value. This process is illustrated in this figure:

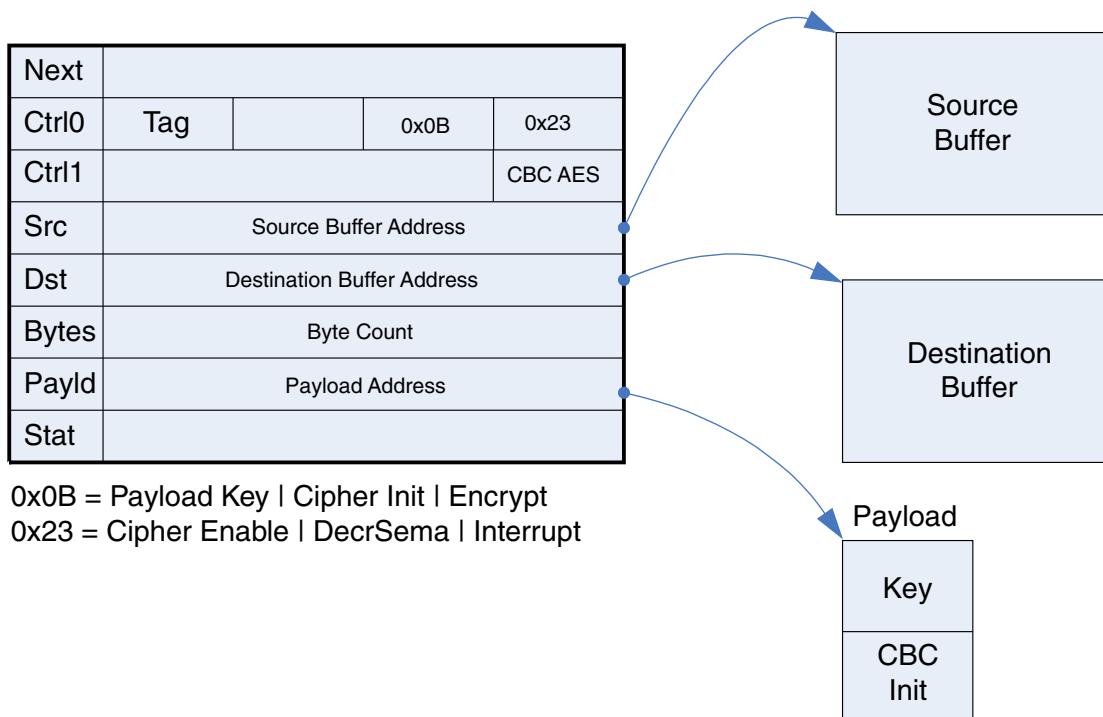


Figure 7-8. Basic cipher operation

```
<code>
typedef struct _dcp_descriptor
{
    u32             *next;
    hw_dcp_packet1_t ctrl0;
    hw_dcp_packet2_t ctrl1;
    u32             *src,
    *dst,
    buf_size,
    *payload,
    stat;
} DCP_DESCRIPTOR;
DCP_DESCRIPTOR dcp1;
```

```

u32 *srcbuffer;
u32 *dstbuffer;
u32 payload[8];
// set up key/CBC init in the payload
payload[0]=0x01234567;           // key
payload[1]=0x89ABCDEF;
payload[2]=0xfedcba98;
payload[3]=0x76543210;
payload[4]=0x00112233;           // CBC initialization
payload[5]=0x44556677;
payload[6]=0x8899aab8;
payload[7]=0xcccddeeff;
// set up control packet
dcp1.next = 0;                  // single packet in chain
dcp1.ctrl0.U = 0;                // clear ctrl0 field
dcp1.ctrl0.B.PAYLOAD_KEY = 1;    // key is located in payload
dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp1.ctrl0.B.CIPHER_INIT = 1;    // init CBC for this block (from payload)
dcp1.ctrl0.B.ENABLE_CIPHER = 1;   // enable cipher
dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp1.ctrl0.B.INTERRUPT = 1;      // interrupt
dcp1.ctrl1.U = 0;                // clear ctrl1
dcp1.ctrl1.B.CIPHER_MODE = 1;    // select CBC mode of operation
dcp1.src = srcbuffer;            // source buffer
dcp1.dst = dstbuffer;            // destination buffer
dcp1.buf_size = 512;             // 512 bytes
dcp1.payload = payload;          // holds key/CBC init
dcp1.status = 0;                 // clear status
// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1);   // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 1);         // increment semaphore by 1
// now wait for interrupt or poll
// polling code
while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>

```

7.3.7.4 Multi-buffer scatter/gather cipher and hash operation programming example

For this example, three separate buffers are encrypted and hashed with the results being directed to a unified buffer (gather operation). Three descriptors are used for the operation because there are three separate source buffer pointers. The DCP reads the data from the source buffer and computes a hash on the unencrypted data. It then encrypts the data and writes it to the destination buffer. For this example, the key is located in the key RAM, and the algorithm uses the AES CBC mode of operation with a SHA-1 hash. The payload for the first operation contains the CBC initialization value, and the payload for the last packet contains the expected hash value. The middle packet requires no payload. This process is illustrated in this figure:

Operation

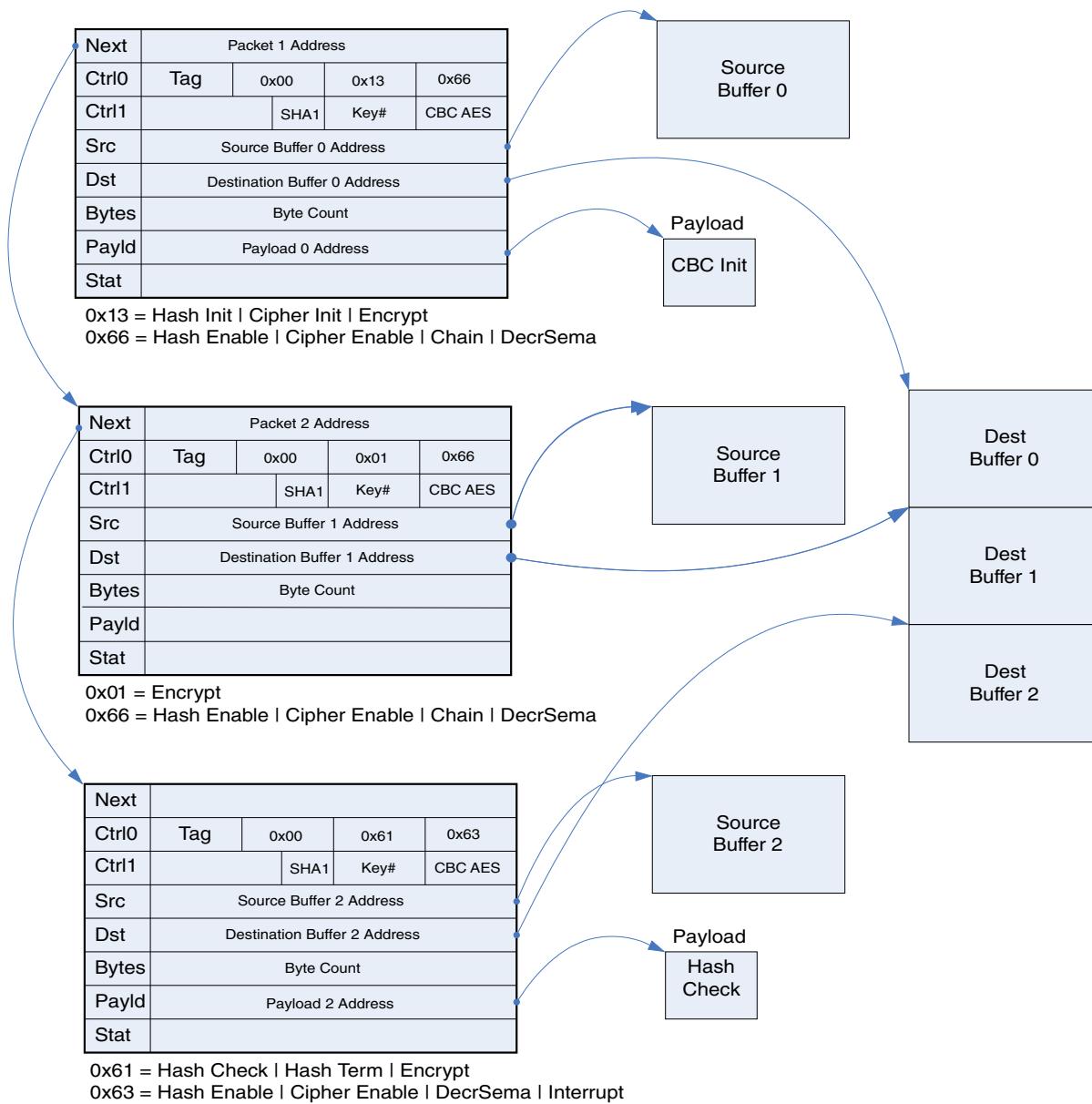


Figure 7-9. Multi-buffer scatter/gather cipher and hash operation

```

<code>
typedef struct _dcp_descriptor
{
    u32             *next;
    hw_dcp_packet1_t ctrl0;
    hw_dcp_packet2_t ctrl1;
    u32             *src,
    *dst,
    buf_size,
    *payload,
    stat;
} DCP_DESCRIPTOR;
DCP_DESCRIPTOR dcp1, dcp2, dcp3;
u32 *srcbuffer0, *srcbuffer2, *srcbuffer3;
u32 *dstbuffer;
u32 payload0[4], payload2[5];
// set up CBC init in the payload
payload0[0]=0x01234567;           // key
payload0[1]=0x89ABCDEF;

```

```

payload0[2]=0xfedcba98;
payload0[3]=0x76543210;
payload2[0]=0x00112233;           // CBC initialization
payload2[1]=0x44556677;
payload2[2]=0x8899aabb;
payload2[3]=0xcccddeff;
payload2[3]=0xaabbccdd;
// set up control packet
dcp1.next = dcp2;                // point to packet 2
dcp1.ctrl0.U = 0;                 // clear ctrl0 field
dcp1.ctrl0.B.CIPHER_ENCRYPT = 1;   // encryption operation
dcp1.ctrl0.B.CIPHER_INIT = 1;      // init CBC for this block (from payload)
dcp1.ctrl0.B.HASH_INIT = 1;        // init hash this block
dcp1.ctrl0.B.ENABLE_CIPHER = 1;    // enable cipher
dcp1.ctrl0.B.ENABLE_HASH = 1;      // enable cipher
dcp1.ctrl0.B.DECR_SEMAPHORE = 1;   // decrement semaphore
dcp1.ctrl0.B.CHAIN = 1;           // chain to next packet
dcp1.ctrl1.U = 0;                 // clear ctrl1
dcp1.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of
operation
dcp1.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1; // select SHA1 hash
dcp1.ctrl1.B.KEY_SELECT = 2;       // select key #2
dcp1.src = srcbuffer0;            // source buffer
dcp1.dst = dstbuffer;             // destination buffer
dcp1.buf_size = 512;              // 512 bytes
dcp1.payload = payload0;          // holds key/CBC init
dcp1.status = 0;                  // clear status
// set up control packet
dcp2.next = dcp3;                // point to packet 2
dcp2.ctrl0.U = 0;                 // clear ctrl0 field
dcp2.ctrl0.B.CIPHER_ENCRYPT = 1;   // encryption operation
dcp2.ctrl0.B.ENABLE_CIPHER = 1;    // enable cipher
dcp2.ctrl0.B.ENABLE_HASH = 1;      // enable cipher
dcp2.ctrl0.B.DECR_SEMAPHORE = 1;   // decrement semaphore
dcp2.ctrl0.B.CHAIN = 1;           // chain to next packet
dcp2.ctrl1.U = 0;                 // clear ctrl1
dcp2.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of
operation
dcp2.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1; // select SHA1 hash
dcp2.ctrl1.B.KEY_SELECT = 2;       // select key #2
dcp2.src = srcbuffer1;            // source buffer
dcp2.dst = dstbuffer+512;          // destination buffer
dcp2.buf_size = 512;              // 512 bytes
dcp2.payload = NULL;              // no payload required
dcp2.status = 0;                  // clear status
// set up control packet
dcp3.next = dcp3;                // point to packet 2
dcp3.ctrl0.U = 0;                 // clear ctrl0 field
dcp3.ctrl0.B.HASH_TERM = 1;        // terminate hash block
dcp3.ctrl0.B.HASH_CHECK = 1;       // check hash against payload value
dcp3.ctrl0.B.CIPHER_ENCRYPT = 1;   // encryption operation
dcp3.ctrl0.B.ENABLE_CIPHER = 1;    // enable cipher
dcp3.ctrl0.B.ENABLE_HASH = 1;      // enable cipher
dcp3.ctrl0.B.DECR_SEMAPHORE = 1;   // decrement semaphore
dcp3.ctrl0.B.INTERRUPT = 1;        // interrupt on completion
dcp3.ctrl1.U = 0;                 // clear ctrl1
dcp3.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of
operation
dcp3.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1; // select SHA1 hash
dcp3.ctrl1.B.KEY_SELECT = 2;       // select key #2
dcp3.src = srcbuffer1;            // source buffer
dcp3.dst = dstbuffer+1024;          // destination buffer
dcp3.buf_size = 512;              // 512 bytes
dcp3.payload = payload2;          // payload is hash check value
dcp3.status = 0;                  // clear status
// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1);    // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 3);          // increment semaphore by 3 (for 3 packets)
// now wait for interrupt or poll
// polling code

```

DCP memory map/register definition

```

while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>

```

7.4 DCP memory map/register definition

DCP hardware register format summary

DCP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
402F_C000	DCP control register 0 (DCP_CTRL)	32	R/W	F080_0000h	7.4.1/241
402F_C010	DCP status register (DCP_STAT)	32	R/W	1000_0000h	7.4.2/243
402F_C020	DCP channel control register (DCP_CHANNELCTRL)	32	R/W	0000_0000h	7.4.3/246
402F_C030	DCP capability 0 register (DCP_CAPABILITY0)	32	R/W	0000_0404h	7.4.4/247
402F_C040	DCP capability 1 register (DCP_CAPABILITY1)	32	R	0007_0001h	7.4.5/248
402F_C050	DCP context buffer pointer (DCP_CONTEXT)	32	R/W	0000_0000h	7.4.6/249
402F_C060	DCP key index (DCP_KEY)	32	R/W	0000_0000h	7.4.7/249
402F_C070	DCP key data (DCP_KEYDATA)	32	R/W	0000_0000h	7.4.8/250
402F_C080	DCP work packet 0 status register (DCP_PACKET0)	32	R	0000_0000h	7.4.9/251
402F_C090	DCP work packet 1 status register (DCP_PACKET1)	32	R	0000_0000h	7.4.10/252
402F_C0A0	DCP work packet 2 status register (DCP_PACKET2)	32	R	0000_0000h	7.4.11/255
402F_C0B0	DCP work packet 3 status register (DCP_PACKET3)	32	R	0000_0000h	7.4.12/256
402F_C0C0	DCP work packet 4 status register (DCP_PACKET4)	32	R	0000_0000h	7.4.13/256
402F_C0D0	DCP work packet 5 status register (DCP_PACKET5)	32	R	0000_0000h	7.4.14/257
402F_C0E0	DCP work packet 6 status register (DCP_PACKET6)	32	R	0000_0000h	7.4.15/257
402F_C100	DCP channel 0 command pointer address register (DCP_CH0CMDPTR)	32	R/W	0000_0000h	7.4.16/258
402F_C110	DCP channel 0 semaphore register (DCP_CH0SEMA)	32	R/W	0000_0000h	7.4.17/259
402F_C120	DCP channel 0 status register (DCP_CH0STAT)	32	R/W	0000_0000h	7.4.18/260
402F_C130	DCP channel 0 options register (DCP_CH0OPTS)	32	R/W	0000_0000h	7.4.19/262
402F_C140	DCP channel 1 command pointer address register (DCP_CH1CMDPTR)	32	R/W	0000_0000h	7.4.20/263
402F_C150	DCP channel 1 semaphore register (DCP_CH1SEMA)	32	R/W	0000_0000h	7.4.21/264
402F_C160	DCP channel 1 status register (DCP_CH1STAT)	32	R/W	0000_0000h	7.4.22/265
402F_C170	DCP channel 1 options register (DCP_CH1OPTS)	32	R/W	0000_0000h	7.4.23/267

Table continues on the next page...

DCP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
402F_C180	DCP channel 2 command pointer address register (DCP_CH2CMDPTR)	32	R/W	0000_0000h	7.4.24/268
402F_C190	DCP channel 2 semaphore register (DCP_CH2SEMA)	32	R/W	0000_0000h	7.4.25/269
402F_C1A0	DCP channel 2 status register (DCP_CH2STAT)	32	R/W	0000_0000h	7.4.26/270
402F_C1B0	DCP channel 2 options register (DCP_CH2OPTS)	32	R/W	0000_0000h	7.4.27/272
402F_C1C0	DCP channel 3 command pointer address register (DCP_CH3CMDPTR)	32	R/W	0000_0000h	7.4.28/273
402F_C1D0	DCP channel 3 semaphore register (DCP_CH3SEMA)	32	R/W	0000_0000h	7.4.29/274
402F_C1E0	DCP channel 3 status register (DCP_CH3STAT)	32	R/W	0000_0000h	7.4.30/275
402F_C1F0	DCP channel 3 options register (DCP_CH3OPTS)	32	R/W	0000_0000h	7.4.31/277
402F_C400	DCP debug select register (DCP_DBGSELECT)	32	R/W	0000_0000h	7.4.32/277
402F_C410	DCP debug data register (DCP_DBGDATA)	32	R	0000_0000h	7.4.33/278
402F_C420	DCP page table register (DCP_PAGETABLE)	32	R/W	0000_0000h	7.4.34/278
402F_C430	DCP version register (DCP_VERSION)	32	R	0201_0000h	7.4.35/279

7.4.1 DCP control register 0 (DCP_CTRL)

The CTRL register contains the controls for the DCP module.

CTRL: 0x000

CTRL_SET: 0x004

CTRL_CLR: 0x008

CTRL_TOG: 0x00C

The control register contains the primary controls for the DCP block. The present bits indicate which sub-features of the block are present in the hardware. The context control bits control how the DCP utilizes its context buffer and the gather residual writes bit controls how the master handles the writing of misaligned data to the bus. Each channel and the color-space converter contain an independent interrupt enable.

EXAMPLE

```
DCP_CTRL_SET(BM_DCP_CTRL_SFTRST);
DCP_CTRL_CLR(BM_DCP_CTRL_SFTRST | BM_DCP_CTRL_CLKGATE);
```

DCP memory map/register definition

Address: 402F_C000h base + 0h offset = 402F_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFRST	CLKGATE	PRESENT_CRYPTO	PRESENT_SHA	Reserved				GATHER_RESIDUAL_WRITES	ENABLE_CONTEXT_CACHING	ENABLE_CONTEXT_SWITCHING	Reserved				
W																
Reset	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								CHANNEL_INTERRUPT_ENABLE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable a normal DCP operation. Set this bit to one (default) to disable the clocking with the DCP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the DCP block to its default state.
30 CLKGATE	This bit must be set to zero for a normal operation. When set to one, it gates off the clocks to the block.
29 PRESENT_CRYPTO	Indicates whether the crypto (cipher/hash) functions are present. 0x1 Present — 0x0 Absent —
28 PRESENT_SHA	Indicates whether the SHA1/SHA2 functions are present. 0x1 Present — 0x0 Absent —
27–24 -	This field is reserved. Reserved, always set to zero
23 GATHER_RESIDUAL_WRITES	The software must set this bit to enable the ragged writes to the unaligned buffers to be gathered between multiple write operations. This improves the performance by removing several byte operations between the write bursts. The trailing byte writes are held in a residual write data buffer and combined with a subsequent write to the buffer to form a word write.
22 ENABLE_CONTEXT_CACHING	The software must set this bit to enable the caching of contexts between the operations. If only a single channel is used for encryption/hashing, enabling the caching causes the context to not be reloaded if the channel was the last one to be used.
21 ENABLE_CONTEXT_SWITCHING	Enable automatic context switching for the channels. The software must set this bit if more than one channel is performing the hashing or cipher operations that require the context to be saved (for example, when the CBC mode is enabled). By disabling the context switching, the software can save the 208 bytes used for the context buffer.
20–9 -	This field is reserved. Reserved, always set to zero
8 RSVD_CSC_INTERRUPT_ENABLE	This field is reserved. Reserved, always set to zero
CHANNEL_INTERRUPT_ENABLE	Per-channel interrupt enable bit. When set, the channel's interrupt is routed to the interrupt controller. Channel 0 is routed to the dcp_vmi_irq signal and the other channels are combined (along with the CRC interrupt) into the dcp_irq signal. 0x01 CH0 — 0x02 CH1 — 0x04 CH2 — 0x08 CH3 —

7.4.2 DCP status register (DCP_STAT)

The DCP interrupt status register provides the channel interrupt status information.

STAT: 0x010

DCP memory map/register definition

STAT_SET: 0x014

STAT_CLR: 0x018

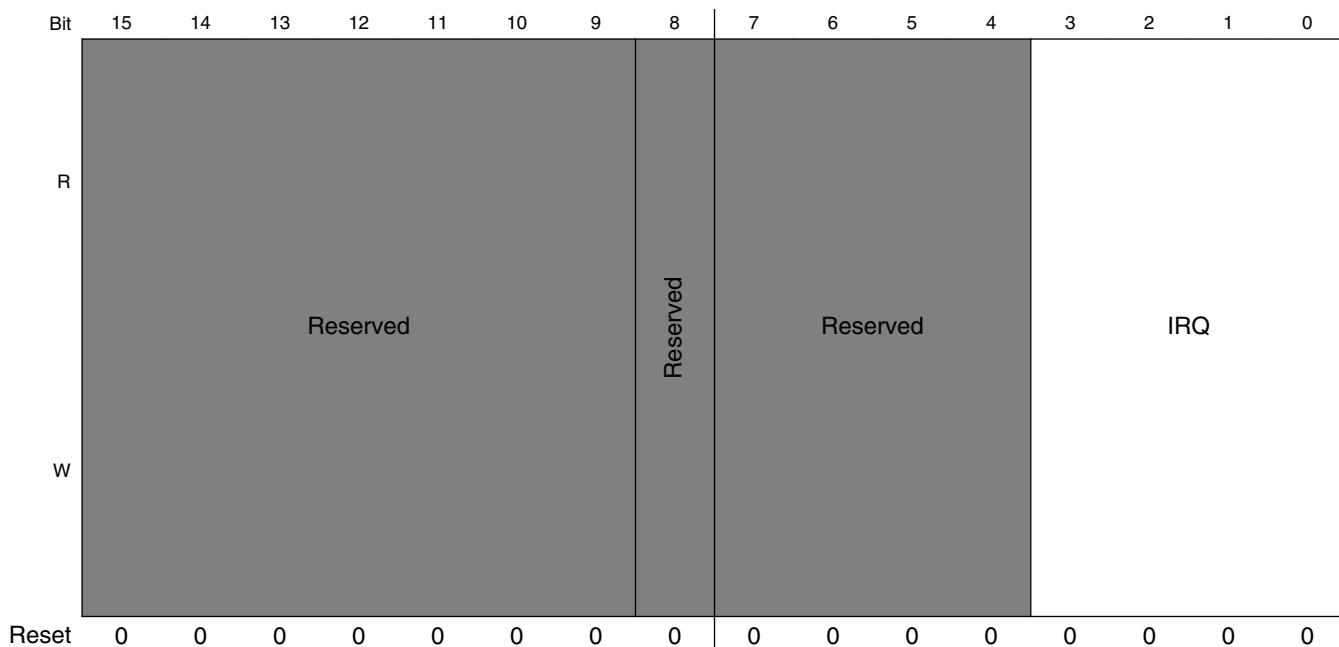
STAT_TOG: 0x01C

This register provides the status feedback indicating which channel is currently performing an operation and which channels have pending operations.

EXAMPLE

Address: 402F_C000h base + 10h offset = 402F_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			OTP_KEY_READY		CUR_CHANNEL							READY_CHANNELS			
W				1	0	0	0	0	0	0	0	0	0	0	0	0

**DCP_STAT field descriptions**

Field	Description
31–29 -	This field is reserved. Reserved, always set to zero
28 OTP_KEY_READY	When set, it indicates that the OTP key is shifted from the fuse block and is ready for use.
27–24 CUR_CHANNEL	Current (active) channel (encoded) 0x0 None — 0x1 CH0 — 0x2 CH1 — 0x3 CH2 — 0x4 CH3 —
23–16 READY_CHANNELS	Indicates which channels are ready to proceed with a transfer (the active channel is also included). Each bit is a one-hot indicating the request status for the associated channel. 0x01 CH0 — 0x02 CH1 — 0x04 CH2 — 0x08 CH3 —
15–9 -	This field is reserved. Reserved, always set to zero
8 RSVD_IRQ	This field is reserved. Reserved, always set to zero
7–4 -	This field is reserved. Reserved, always set to zero
IRQ	Indicates which channels have pending interrupt requests. The channel 0 interrupt is routed through the dcp_vmi_irq and the other interrupt bits are routed through the dcp_irq.

7.4.3 DCP channel control register (DCP_CHANNELCTRL)

The DCP channel control register provides the controls for the channel arbitration and channel enables.

CHANNELCTRL: 0x020

CHANNELCTRL_SET: 0x024

CHANNELCTRL_CLR: 0x028

CHANNELCTRL_TOG: 0x02C

This register provides the status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

```
BW_DCP_CHANNELCTRL_ENABLE_CHANNEL(BV_DCP_CHANNELCTRL_ENABLE_CHANNEL__CH0); //  
enable channel 0
```

Address: 402F_C000h base + 20h offset = 402F_C020h

Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16	
R	Reserved														CH0_IRQ_MERGED			
W																		
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		
Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0	
R	HIGH_PRIORITY_CHANNEL									ENABLE_CHANNEL								
W																		
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	

DCP_CHANNELCTRL field descriptions

Field	Description
31–17 RSVD	This field is reserved. Reserved, always set to zero
16 CH0_IRQ_MERGED	Indicates that the interrupt for channel 0 must be merged with the other interrupts on the shared dcp_irq interrupt. When set to 0, the channel 0 interrupt is routed to the separate dcp_vmi_irq. When set to 1, the interrupt is routed to the shared DCP interrupt.

Table continues on the next page...

DCP_CHANNELCTRL field descriptions (continued)

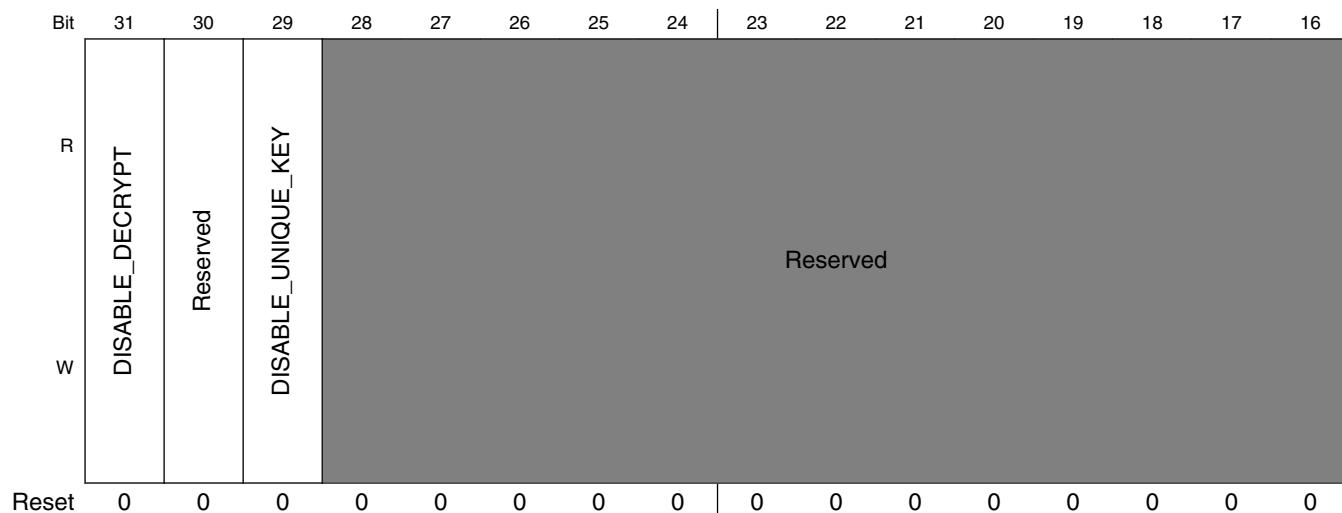
Field	Description
15–8 HIGH_PRIORITY_CHANNEL	Setting a bit in this field causes the corresponding channel to have high-priority arbitration. The high-priority channels are arbitrated round-robin and take the precedence over other channels that are not marked as high-priority. 0x01 CH0 — 0x02 CH1 — 0x04 CH2 — 0x08 CH3 —
ENABLE_CHANNEL	Setting a bit in this field enables the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When not enabled, the channel is denied access to the central DMA resources. 0x01 CH0 — 0x02 CH1 — 0x04 CH2 — 0x08 CH3 —

7.4.4 DCP capability 0 register (DCP_CAPABILITY0)

This register contains additional information about the DCP module implementation parameters.

This register provides the capability information for the DCP block. It indicates the number of channels implemented, as well as the number of the key storage locations available for the software use.

Address: 402F_C000h base + 30h offset = 402F_C030h



DCP memory map/register definition

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

DCP CAPABILITY0 field descriptions

Field	Description
31 DISABLE_ DECRYPT	Write to 1 to disable the decryption. This bit can only be written by the secure software and the value can only be cleared by a reset.
30 Reserved	This field is reserved. -
29 DISABLE_ UNIQUE_KEY	Write to a 1 to disable the per-device unique key. The device-specific hardware key may be selected by using a value of 0xFE in the key-select field.
28-12 RSVD	This field is reserved. Reserved, always set to zero
11-8 NUM_ CHANNELS	Encoded value indicating the number of channels implemented in the design
NUM_KEYS	Encoded value indicating the number of key-storage locations implemented in the design

7.4.5 DCP capability 1 register (DCP_CAPABILITY1)

This register contains the information about the algorithms available in the implementation.

This register provides the capability information for the DCP block. It contains two fields indicating which encryption and hashing algorithms are present in the design. Each bit set indicates that the support for the associated function is present.

Address: 402F_C000h base + 40h offset = 402F_C040h

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

DCP_CAPABILITY1 field descriptions

Field	Description
31–16 HASH_ALGORITHMS	One-hot field indicating which hashing features are implemented in the hardware 0x0001 SHA1 — 0x0002 CRC32 — 0x0004 SHA256 —
CIPHER_ALGORITHMS	One-hot field indicating which cipher algorithms are available 0x0001 AES128 —

7.4.6 DCP context buffer pointer (DCP_CONTEXT)

This register contains a pointer to the memory region to be used for the DCP context swap operations.

This register contains a pointer to the start of the context pointer memory in the on-chip SRAM or off-chip SDRAM. This buffer is used to store the state information when the DCP module changes from one channel to another.

Address: 402F_C000h base + 50h offset = 402F_C050h

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

DCP_CONTEXT field descriptions

Field	Description
ADDR	Context pointer address. The address must be located in the system RAM and must be word-aligned for the optimal performance.

7.4.7 DCP key index (DCP_KEY)

This register contains a pointer to the key location to be written.

DCP memory map/register definition

The DCP module maintains a set of write-only keys that can be used by the software. To write a key, the software must first write the desired key index/subword to this register and then write the key values to the key registers (below). After each write to the key data register, the SUBWORD field increments to enable the software to write the subsequent key to be written without having to rewrite the key index.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeef
DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0)); // set key index to key 0, subword
0
DCP_KEYDATA_WR(0xcccddeeef); // write key values (subword 0)
DCP_KEYDATA_WR(0x8899aabb); // write key values (subword 1)
DCP_KEYDATA_WR(0x44556677); // write key values (subword 2)
DCP_KEYDATA_WR(0x00112233); // write key values (subword 3)
```

Address: 402F_C000h base + 60h offset = 402F_C060h

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

DCP_KEY field descriptions

Field	Description
31–8 RSVD	This field is reserved. Reserved, always set to zero
7–6 RSVD_INDEX	This field is reserved. Reserved, always set to zero
5–4 INDEX	Key index pointer. The valid indices are 0-[number_keys].
3–2 RSVD_SUBWORD	This field is reserved. Reserved always set to zero
SUBWORD	Key subword pointer. The valid indices are 0-3. After each write to the key data register, this field increments.

7.4.8 DCP key data (DCP_KEYDATA)

This register provides the write access to the key/key subword specified by the key index register.

Writing this location updates the selected subword for the key located at the index specified by the key index register. The write also triggers the SUBWORD field of the KEY register to increment to the next higher word in the key.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeef
DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0)); // set key index to key 0, subword
0
DCP_KEYDATA_WR(0xcccddeeef); // write key values (subword 0)
DCP_KEYDATA_WR(0x8899aabb); // write key values (subword 1)
DCP_KEYDATA_WR(0x44556677); // write key values (subword 2)
DCP_KEYDATA_WR(0x00112233); // write key values (subword 3)
```

Address: 402F_C000h base + 70h offset = 402F_C070h

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

DCP_KEYDATA field descriptions

Field	Description
DATA	Word 0 data for the key. This is the least-significant word.

7.4.9 DCP work packet 0 status register (DCP_PACKET0)

This register displays the values for the current work packet offset 0x00 (Next Command) field.

The work packet status registers show the contents of the currently executing packet. When the channels are inactive, the packet status register returns 0. The register bits are fully listed here to document the packet structure in the memory.

Address: 402F_C000h base + 80h offset = 402F_C080h

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

DCP_PACKET0 field descriptions

Field	Description
ADDR	Next pointer register

7.4.10 DCP work packet 1 status register (DCP_PACKET1)

This register displays the values for the current work packet offset 0x04 (control) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: 402F_C000h base + 90h offset = 402F_C090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWAP	TEST_SEMA_IRQ	CONSTANT_FILL
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HASH_OUTPUT	CHECK_HASH	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYPT	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHER	ENABLE_MEMCOPY	CHAIN_CONTIGUOUS	CHAIN	DECR_SEMAPHORE	INTERRUPT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET1 field descriptions

Field	Description
31–24 TAG	Packet Tag
23 OUTPUT_WORDSWAP	Reflects whether the DCP engine wordswaps the output data (big-endian data).
22 OUTPUT_BYTESWAP	Reflects whether the DCP engine byteswaps the output data (big-endian data).
21 INPUT_WORDSWAP	Reflects whether the DCP engine wordswaps the input data (big-endian data).
20 INPUT_BYTESWAP	Reflects whether the DCP engine byteswaps the input data (big-endian data).
19 KEY_WORDSWAP	Reflects whether the DCP engine swaps the key words (big-endian key).
18 KEY_BYTESWAP	Reflects whether the DCP engine swaps the key bytes (big-endian key).

Table continues on the next page...

DCP_PACKET1 field descriptions (continued)

Field	Description
17 TEST_SEMA_IRQ	This bit is used to test the channel semaphore transition to 0. FOR TEST USE ONLY!
16 CONSTANT_FILL	When this bit is set (MEMCOPY and BLIT modes only), the DCP simply fills the destination buffer with the value found in the source address field.
15 HASH_OUTPUT	When the hashing is enabled, this bit controls whether the input or output data is hashed. 0 INPUT — 1 OUTPUT —
14 CHECK_HASH	Reflects whether the calculated hash value must be compared to the hash provided in the payload.
13 HASH_TERM	Reflects whether the current hashing block is the final block in the hashing operation, so the hash padding must be applied by the hardware.
12 HASH_INIT	Reflects whether the current hashing block is the initial block in the hashing operation, so the hash registers must be initialized before the operation.
11 PAYLOAD_KEY	When set, it indicates the payload contains the key. This bit takes precedence over the OTP_KEY control.
10 OTP_KEY	Reflects whether a hardware-based key must be used. The KEY_SELECT field from the Control1 field is used to select from multiple hardware keys. The PAYLOAD_KEY bit takes precedence over the OTP_KEY bit.
9 CIPHER_INIT	Reflects whether the cipher block must load the initialization vector from the payload for this operation.
8 CIPHER_ENCRYPT	When the cipher block is enabled, this bit indicates whether the operation is encryption or decryption. 1 ENCRYPT — 0 DECRYPT —
7 ENABLE_BLIT	Reflects whether the DCP must perform a blit operation. The source data is always continuous and the destination buffer is written in the run/stride format. When set, the BUFFER_SIZE field is treated as two 16-bit values for the X-Y extents of the blit operation.
6 ENABLE_HASH	Reflects whether the selected hashing function must be enabled for this operation.
5 ENABLE_CIPHER	Reflects whether the selected cipher function must be enabled for this operation.
4 ENABLE_MEMCOPY	Reflects whether the selected hashing function should be enabled for this operation.
3 CHAIN_CONTIGUOUS	Reflects whether the next packet's address is located following this packet's payload.
2 CHAIN	Reflects whether the next command pointer register must be loaded into the channel's current descriptor pointer.
1 DECR_SEMAPHORE	Reflects whether the channel's semaphore must be decremented at the end of the current operation. When the semaphore reaches a value of zero, no more operations are issued from the channel.
0 INTERRUPT	Reflects whether the channel must issue an interrupt upon the completion of the packet.

7.4.11 DCP work packet 2 status register (DCP_PACKET2)

This register displays the values for the current work packet offset 0x08 (Control1) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: 402F_C000h base + A0h offset = 402F_C0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CIPHER_CFG								Reserved				HASH_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KEY_SELECT								CIPHER_MODE				CIPHER_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET2 field descriptions

Field	Description
31–24 CIPHER_CFG	Cipher configuration bits. Optional configuration bits are required for the ciphers.
23–20 RSVD	This field is reserved. Reserved, always set to zero
19–16 HASH_SELECT	Hash Selection Field 0x00 SHA1 — 0x01 CRC32 — 0x02 SHA256 —
15–8 KEY_SELECT	Key selection field. The value here reflects the key index for the cipher operation. Values 0-3 refer to the software keys that can be written to the key RAM. The OTP key or the unique device-specific key may also be selected with a value of 0xFF (OTP key) or 0xFE (unique key). 0x00 KEY0 — 0x01 KEY1 — 0x02 KEY2 — 0x03 KEY3 — 0xFE UNIQUE_KEY — 0xFF OTP_KEY —
7–4 CIPHER_MODE	Cipher mode selection field. Reflects the mode of operation for the cipher operations. 0x00 ECB — 0x01 CBC —
CIPHER_SELECT	Cipher selection field 0x00 AES128 —

7.4.12 DCP work packet 3 status register (DCP_PACKET3)

This register displays the values for the current work packet offset 0x0C (source address) field.

This register shows the contents of the source address register from the packet being processed. When the CONSTANT_FILL bit in the Control 0 field is set, this field contains the data written to the destination buffer.

Address: 402F_C000h base + B0h offset = 402F_C0B0h

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

DCP_PACKET3 field descriptions

Field	Description
ADDR	Source buffer address pointer. This value is the working value and updates as the operation proceeds.

7.4.13 DCP work packet 4 status register (DCP_PACKET4)

This register displays the values for the current work packet offset 0x10 (destination address) field.

This register shows the contents of the destination address register from the packet being processed.

Address: 402F_C000h base + C0h offset = 402F_C0C0h

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

DCP_PACKET4 field descriptions

Field	Description
ADDR	Destination buffer address pointer. This value is the working value and updates as the operation proceeds.

7.4.14 DCP work packet 5 status register (DCP_PACKET5)

This register displays the values for the current work packet offset 0x14 (buffer size) field.

This register shows the contents of the bytecount register from the packet being processed. The field can be considered either a byte count or a buffer size. The logic treats this as a decrementing count of bytes from the buffer size programmed into the field. As the transaction proceeds, the logic decrements the bytecount as the data is written into the destination buffer. For blit operations, the top 16 bits of this field represent the number of lines (y size) in the blit and the lower 16 bits represent the number of bytes in a line (x size).

Address: 402F_C000h base + D0h offset = 402F_C0D0h

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

DCP_PACKET5 field descriptions

Field	Description
COUNT	Byte count register. This value is the working value and updates as the operation proceeds.

7.4.15 DCP work packet 6 status register (DCP_PACKET6)

This register displays the values for the current work packet offset 0x1C (payload pointer) field.

This register shows the contents of the payload pointer field from the packet being processed.

Address: 402F_C000h base + E0h offset = 402F_C0E0h

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

DCP_PACKET6 field descriptions

Field	Description
ADDR	This register reflects the payload pointer for the current control packet.

7.4.16 DCP channel 0 command pointer address register (DCP_CH0CMDPTR)

The DCP channel 0 current command addresses the register points to the multiword descriptor that is to be executed (or is currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in the memory and then updating the semaphore to a non-zero value. After the engine completes the processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable the processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for access to the engine for the subsequent operation.

DCP channel 0 is controlled by a variable-sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
DCP_CHnCMDPTR_WR(0, v); // Write channel 0 command pointer
pCurptr = (DCP_chncmdptr_t *) DCP_CHnCMDPTR_RD(0); // Read current command
pointer
```

Address: 402F_C000h base + 100h offset = 402F_C100h

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

Reset 0

DCP_CH0CMDPTR field descriptions

Field	Description
ADDR	Pointer to the descriptor structure to be processed for channel 0.

7.4.17 DCP channel 0 semaphore register (DCP_CH0SEMA)

The DCP channel 0 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain is generated in the memory, the software must write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor must be loaded into the channel upon the completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore must be decremented after the operation. A channel is considered active when the semaphore is of a non-zero value. When programming a series of operations, the software must properly program the semaphore values in conjunction with the "decrement_semaphore" bits in the control packets to ensure that a proper number of descriptors is activated. A semaphore may be cleared by the software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error occurs.

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. After processing each control packet, the DCP decrements the semaphore if it is non-zero. The channel continues to process the packets as long as the semaphore contains a non-zero value and the CHAIN or CHAIN_CONTIGOUS control bits in the Control0 field are set.

Address: 402F_C000h base + 110h offset = 402F_C110h

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

DCP_CH0SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero
INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that the simultaneous software adds and DCP hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two (unless the DCP channel decrements the count on the same clock) and then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF into the DCP_CHnSEMA_CLR register.

7.4.18 DCP channel 0 status register (DCP_CH0STAT)

The DCP channel 0 interrupt status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

DCP CHOSTAT: 0x120

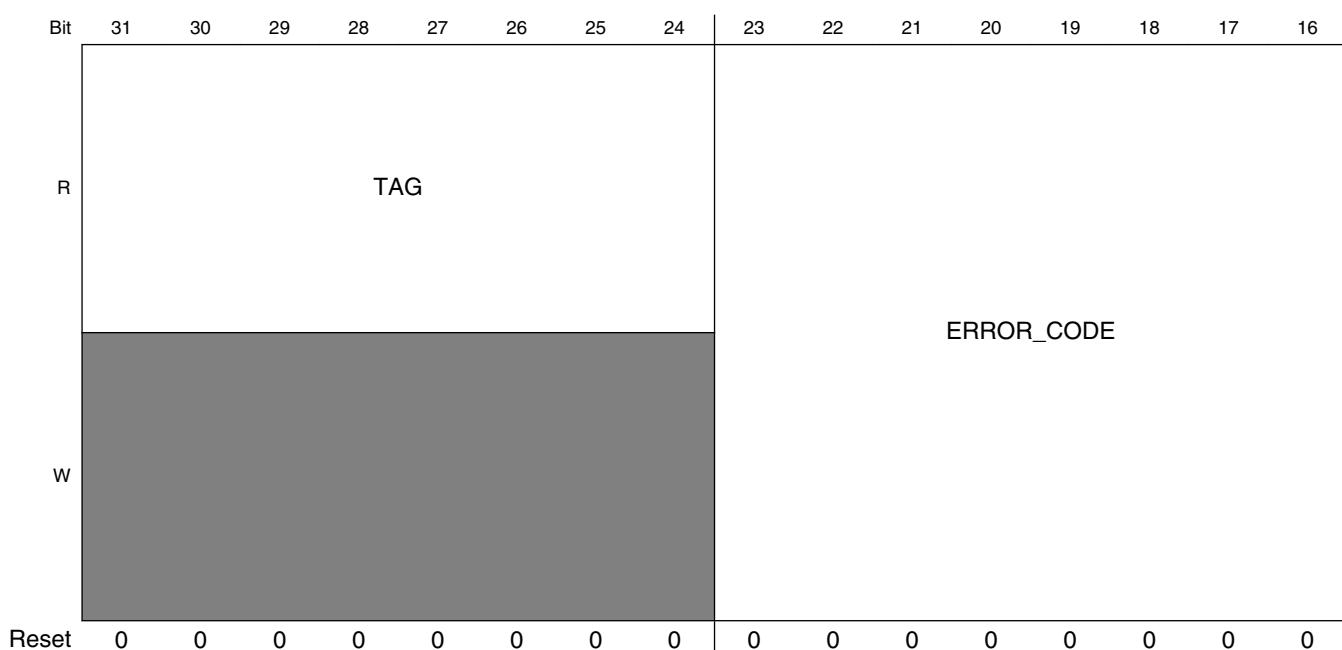
CH0STAT SET: 0x124

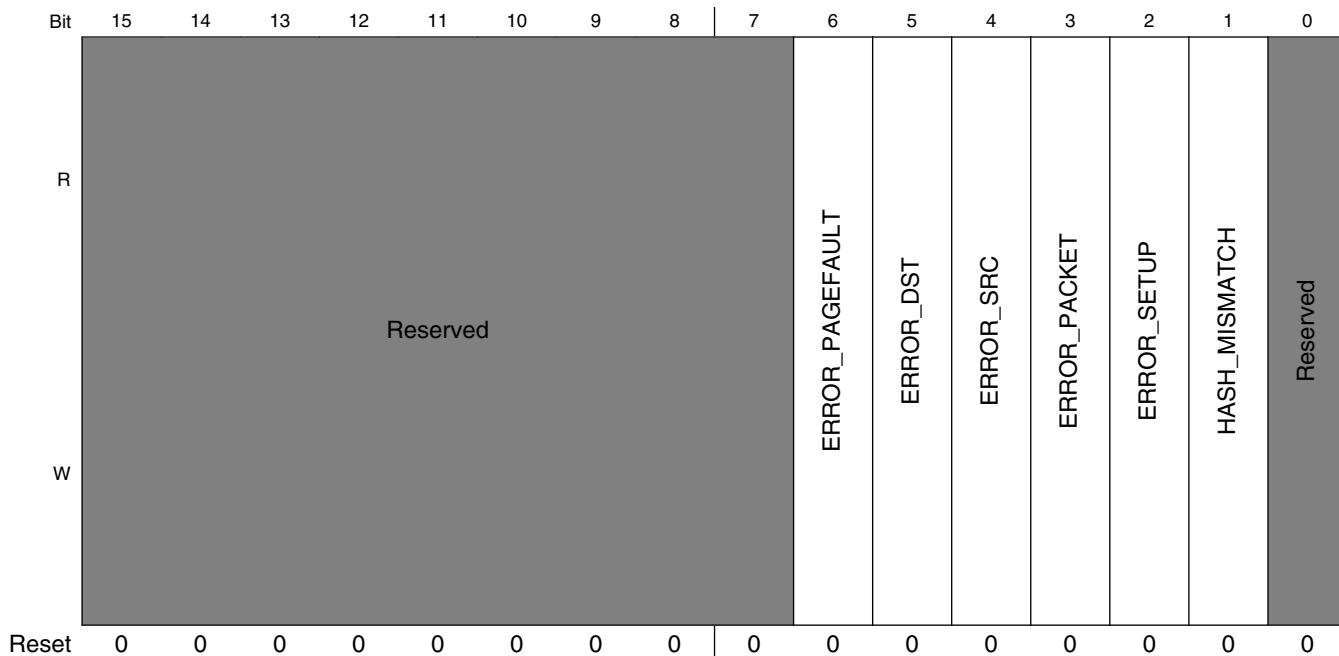
CH0STAT CLR: 0x128

CH0STAT TOG: 0x12C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated when the packet is completed. In addition, the tag value from the command is stored in the TAG field so that the software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and the processing of the command chain is halted.

Address: 402F C000h base + 120h offset = 402F C120h



**DCP_CH0STAT field descriptions**

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates the additional error codes for some of the error conditions 0x01 NEXT_CHAIN_IS_0 — Error signalled because the next pointer is 0x00000000 0x02 NO_CHAIN — Error signalled because the semaphore is non-zero and neither chain bit is set 0x03 CONTEXT_ERROR — Error signalled because an error is reported reading/writing the context buffer 0x04 PAYOUTLOAD_ERROR — Error signalled because an error is reported reading/writing the payload 0x05 INVALID_MODE — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	This field is reserved. Reserved, always set to zero
6 ERROR_PAGEFAULT	This bit indicates that a page fault occurred while converting a virtual address to a physical address. When an error is detected, the channel's processing stops until the error is handled by the software.
5 ERROR_DST	This bit indicates that a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
4 ERROR_SRC	This bit indicates that a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
3 ERROR_PACKET	This bit indicates that a bus error occurred when reading the packet or payload, or when writing the status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by the software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration (such as a buffer length that is not a multiple of the natural data size for the operation). When an error is detected, the channel's processing stops until the error is handled by the software.

Table continues on the next page...

DCP CH0STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	This bit indicates that a hashing check operation mismatched for the control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by the software.
0 RSVD_ COMPLETE	This field is reserved. This bit always reads 0 in the status register, but it is set to 1 in the packet status field after the processing of the packet completes. This is done so that the software can verify that each packet completed properly in a chain of commands for the cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

7.4.19 DCP channel 0 options register (DCP_CH0OPTS)

The DCP channel 0 options status register contains optional control information that may be used to further tune the behavior of the channel.

DCP CHOOPTS: 0x130

CHOOPTS SET: 0x134

CH0OPTS CLR: 0x138

CH0OPTS TOG: 0x13C

The options register can be used to control the optional features of the channels.

Address: 402F C000h base + 130h offset = 402F C130h

DCP CHOOPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero
RECOVERY_TIMER	This field indicates the recovery time for the channel. After each operation, the recovery timer for the channel is initialized with this value and then decremented until the timer reaches zero. The channel does not initiate another operation for the next packet in the chain until the recovery time is satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range from 0 ns to 8.3 ms at the operation frequency of 133 MHz.

7.4.20 DCP channel 1 command pointer address register (DCP_CH1CMDPTR)

The DCP channel 1 current command address register points to the multiword descriptor that is to be executed (or is currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in the memory and then updating the semaphore to a non-zero value. After the engine completes the processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable the processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for the access to the engine for the subsequent operation.

DCP channel 1 is controlled by a variable-sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
DCP_CHn_CMDPTR_WR(1, v); // Write channel 1 command pointer
pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(1); // Read current command
pointer
```

Address: 402F_C000h base + 140h offset = 402F_C140h

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

DCP_CH1CMDPTR field descriptions

Field	Description
ADDR	Pointer to the descriptor structure to be processed for channel 1.

7.4.21 DCP channel 1 semaphore register (DCP_CH1SEMA)

The DCP channel 1 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain is generated in the memory, the software must write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor must be loaded into the channel upon the completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore must be decremented after the operation. A channel is considered active when the semaphore is of a non-zero value. When programming a series of operations, the software must properly program the semaphore values in conjunction with the "decrement_semaphore" bits in the control packets to ensure that a proper number of descriptors is activated. A semaphore may be cleared by the software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error occurs.

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. The DCP processing continues until the engine attempts to decrement a semaphore that already reached a value of zero. When the attempt is made, the DCP channel is stalled until the software increments the semaphore count.

Address: 402F_C000h base + 150h offset = 402F_C150h

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

DCP_CH1SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero
INCREMENT	The value written to this field is added to the semaphore count in an atomic way, such that the simultaneous software adds and the DCP hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two (when the DCP channel decrements the count on the same clock then the count is

Table continues on the next page...

DCP_CH1SEMA field descriptions (continued)

Field	Description
	incremented by a net one). The semaphore may be cleared by writing 0xFF into the DCP_CHnSEMA_CLR register.

7.4.22 DCP channel 1 status register (DCP_CH1STAT)

The DCP channel 1 interrupt status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

CH1STAT: 0x160

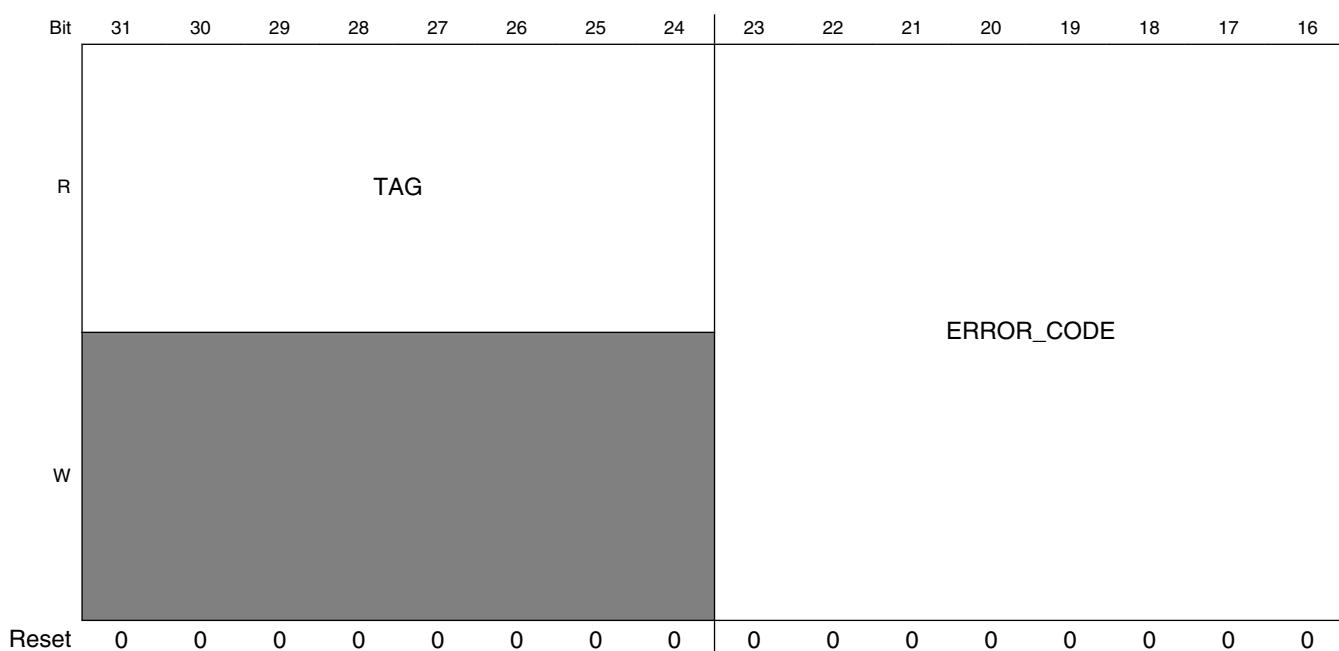
CH1STAT_SET: 0x164

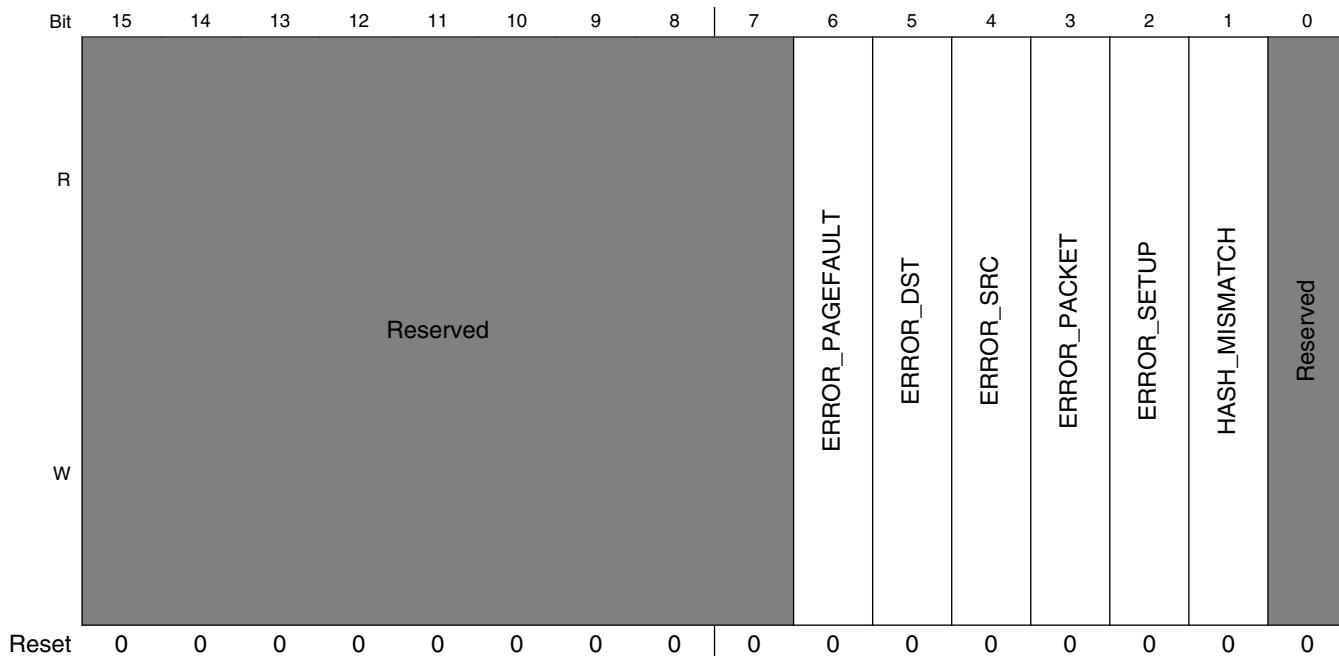
CH1STAT_CLR: 0x168

CH1STAT_TOG: 0x16C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated when the packet completes. In addition, the tag value from the command is stored in the TAG field so that the software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: 402F_C000h base + 160h offset = 402F_C160h



**DCP_CH1STAT field descriptions**

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure.
23–16 ERROR_CODE	Indicates the additional error codes for some of the error conditions. 0x01 NEXT_CHAIN_IS_0 — Error is signalled because the next pointer is 0x00000000. 0x02 NO_CHAIN — Error is signalled because the semaphore is of a non-zero value and neither of the chain bits is set. 0x03 CONTEXT_ERROR — Error is signalled because an error was reported when reading/writing the context buffer. 0x04 PAYOUTLOAD_ERROR — Error is signalled because an error was reported when reading/writing the payload. 0x05 INVALID_MODE — Error is signalled because the control packet specifies an invalid mode select (for example, blit + hash).
15–7 -	This field is reserved. Reserved, always set to zero
6 ERROR_PAGEFAULT	This bit indicates that a page fault occurred while converting a virtual address to a physical address. When an error is detected, the channel's processing stops until the error is handled by the software.
5 ERROR_DST	This bit indicates that a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
4 ERROR_SRC	This bit indicates that a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
3 ERROR_PACKET	This bit indicates that a bus error occurred when reading the packet or payload, or when writing the status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by the software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration (such as a buffer length that is not a multiple of the natural data size for the operation). When an error is detected, the channel's processing stops until the error is handled by the software.

Table continues on the next page...

DCP CH1STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	This bit indicates that a hashing check operation is mismatched for the control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by the software.
0 RSVD_ COMPLETE	This field is reserved. This bit always reads 0 in the status register, but is set to 1 in the packet status field after the processing of the packet completes. This is done so that the software can verify that each packet completed properly in a chain of commands for the cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

7.4.23 DCP channel 1 options register (DCP_CH1OPTS)

The DCP channel 1 options status register contains the optional control information that may be used to further tune the behavior of the channel.

DCP CH1OPTS: 0x170

DCP CH1OPTS SET: 0x174

CH1OPTS CLR: 0x178

CH1OPTS TOG: 0x17C

The options register can be used to control the optional features of the channels.

Address: 402E C000h base + 170h offset = 402E C170h

DCP CH1OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero
RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recovery timer for the channel is initialized with this value and then decremented until the timer reaches zero. The channel does not initiate the operation on the next packet in the chain until the recovery time expires. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range from 0 ns to 8.3 ms at the operation frequency of 133 MHz.

7.4.24 DCP channel 2 command pointer address register (DCP_CH2CMDPTR)

The DCP channel 2 current command address register points to the multiword descriptor that is to be executed (or is currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in the memory and then updating the semaphore to a non-zero value. After the engine completes the processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable the processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for the access to the engine for the subsequent operation.

DCP channel 2 is controlled by a variable-sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
DCP_CHn_CMDPTR_WR(2, v); // Write channel 2 command pointer
pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(2); // Read current command
pointer
```

Address: 402F_C000h base + 180h offset = 402F_C180h

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

DCP_CH2CMDPTR field descriptions

Field	Description
ADDR	Pointer to the descriptor structure to be processed for channel 2.

7.4.25 DCP channel 2 semaphore register (DCP_CH2SEMA)

The DCP channel 2 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain is generated in the memory, the software must write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor must be loaded into the channel upon the completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit which indicates that the counting semaphore must be decremented after the operation. A channel is considered active when the semaphore is of a non-zero value. When programming a series of operations, the software must properly program the semaphore values in conjunction with the "decrement_semaphore" bits in the control packets to ensure that a proper number of descriptors is activated. A semaphore may be cleared by the software by writing 0xFF into the DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error occurs.

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. The DCP processing continues until the engine attempts to decrement a semaphore that already reached a value of zero. When the attempt is made, the DCP channel is stalled until the software increments the semaphore count.

Address: 402F_C000h base + 190h offset = 402F_C190h

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

DCP_CH2SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero
INCREMENT	The value written to this field is added to the semaphore count in an atomic way, such that the simultaneous software adds and DCP hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two. When the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF into the DCP_CHnSEMA_CLR register.

7.4.26 DCP channel 2 status register (DCP_CH2STAT)

The DCP channel 2 interrupt status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

CH2STAT: 0x1A0

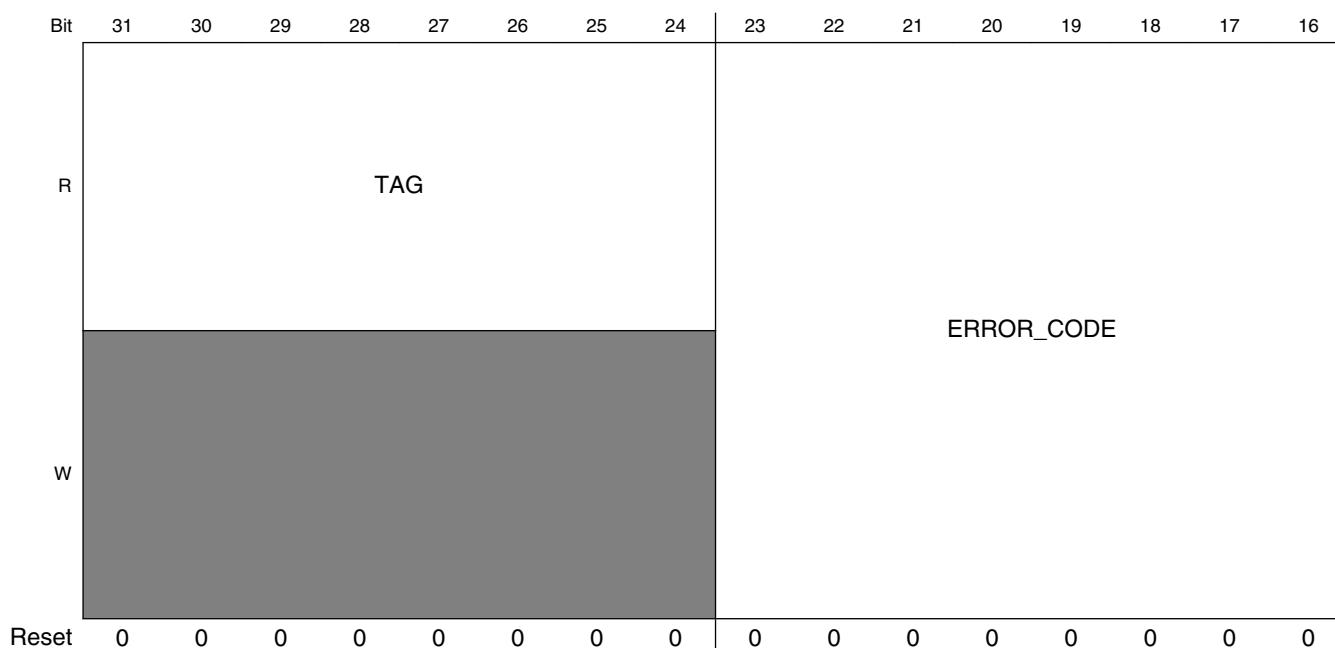
CH2STAT SET: 0x1A4

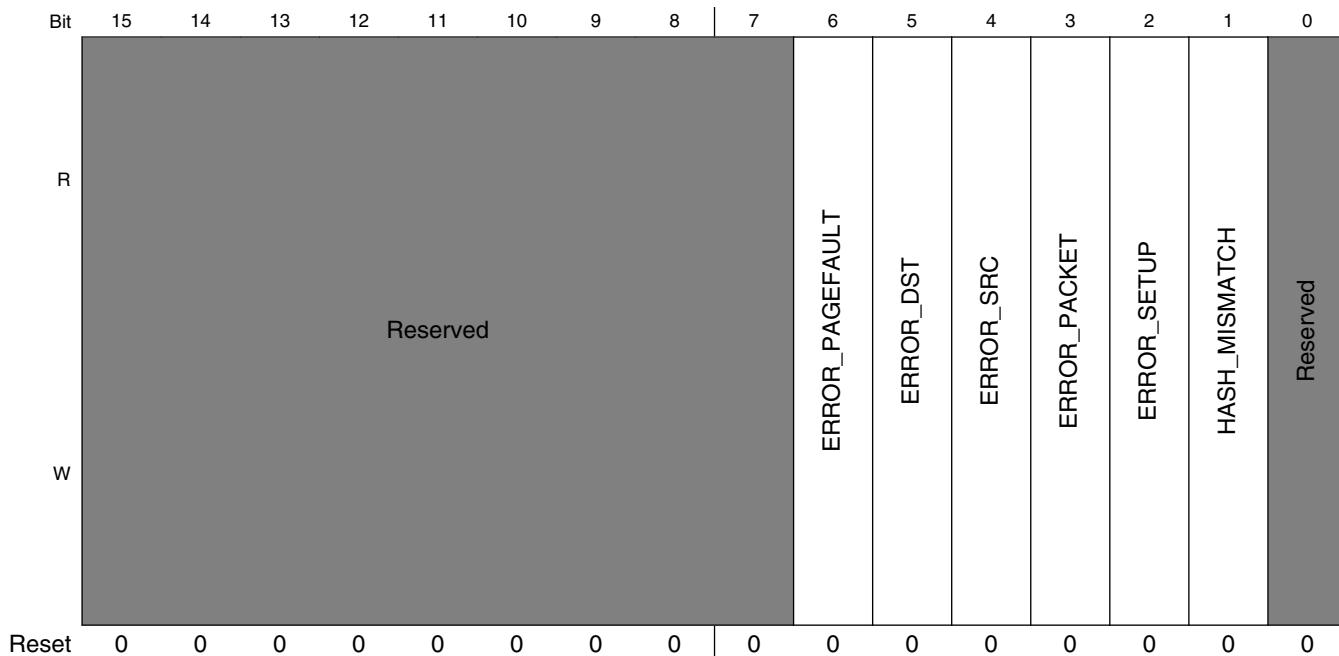
CH2STAT CLR: 0x1A8

CH2STAT TOG: 0x1AC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated when the packet completes. In addition, the tag value from the command is stored in the TAG field, so that the software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and the processing of the command chain is halted.

Address: 402F C000h base + 1A0h offset = 402F C1A0h



**DCP_CH2STAT field descriptions**

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure.
23–16 ERROR_CODE	<p>Indicates additional error codes for some of the error conditions.</p> <ul style="list-style-type: none"> 0x01 NEXT_CHAIN_IS_0 — Error is signalled because the next pointer is 0x00000000. 0x02 NO_CHAIN — Error is signalled because the semaphore is of a non-zero value and neither of the chain bits is set. 0x03 CONTEXT_ERROR — Error is signalled because an error was reported while reading/writing the context buffer. 0x04 PAYOUTLOAD_ERROR — Error is signalled because an error was reported while reading/writing the payload. 0x05 INVALID_MODE — Error is signalled because the control packet specifies an invalid mode select (for instance, blit + hash).
15–7 -	This field is reserved. Reserved, always set to zero
6 ERROR_PAGEFAULT	This bit indicates that a page fault occurred while converting a virtual address to a physical address. When an error is detected, the channel's processing stops until the error is handled by the software.
5 ERROR_DST	This bit indicates that a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
4 ERROR_SRC	This bit indicates that a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
3 ERROR_PACKET	This bit indicates that a bus error occurred when reading the packet or payload, or when writing the status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by the software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration (such as a buffer length that is not a multiple of the natural data size for the operation). When an error is detected, the channel's processing stops until the error is handled by the software.

Table continues on the next page...

DCP CH2STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	This bit indicates that a hashing check operation is mismatched for the control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by the software.
0 RSVD_ COMPLETE	This field is reserved. This bit always reads 0 in the status register, but is set to 1 in the packet status field after the processing of the packet completes. This is done so that the software can verify that each packet completed properly in a chain of commands for the cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

7.4.27 DCP channel 2 options register (DCP_CH2OPTS)

The DCP channel 2 options status register contains optional control information that can be used to further tune the behavior of the channel.

CH2OPTS: 0x1B0

CH2OPTS SET: 0x1B4

CH2OPTS CLR: 0x1B8

CH2OPTS TOG: 0x1BC

The options register can be used to control the optional features of the channels.

Address: 402E C000h base + 1B0h offset = 402E C1B0h

DCP CH2OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero
RECOVERY_TIMER	This field indicates the recovery time for the channel. After each operation, the recovery timer for the channel is initialized with this value and then decremented until the timer reaches zero. The channel does not initiate the operation on the next packet in the chain until the recovery time expires. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range from 0 ns to 8.3 ms at an operation frequency of 133 MHz.

7.4.28 DCP channel 3 command pointer address register (DCP_CH3CMDPTR)

The DCP channel 3 current command address register points to the multiword descriptor that is to be executed (or is currently being executed). The channel can be activated by writing the command pointer address to a valid descriptor in the memory and then updating the semaphore to a non-zero value. After the engine completes the processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable the processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for the access to the engine for the subsequent operation.

DCP channel 3 is controlled by a variable-sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
DCP_CHn_CMDPTR_WR(3, v); // Write channel 3 command pointer
pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(3); // Read current command
pointer
```

Address: 402F_C000h base + 1C0h offset = 402F_C1C0h

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

DCP_CH3CMDPTR field descriptions

Field	Description
ADDR	Pointer to the descriptor structure to be processed for channel 3.

7.4.29 DCP channel 3 semaphore register (DCP_CH3SEMA)

The DCP channel 3 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain is generated in the memory, the software must write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor must be loaded into the channel upon the completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit which indicates that the counting semaphore must be decremented after the operation. A channel is considered active when the semaphore is of a non-zero value. When programming a series of operations, the software must properly program the semaphore values in conjunction with the "decrement_semaphore" bits in the control packets to ensure that a proper number of descriptors is activated. A semaphore can be cleared by the software by writing 0xFF into the DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error occurs.

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. The DCP processing continues until the engine attempts to decrement a semaphore that already reached a value of zero. When the attempt is made, the DCP channel is stalled until the software increments the semaphore count.

Address: 402F_C000h base + 1D0h offset = 402F_C1D0h

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

DCP_CH3SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero
INCREMENT	The value written to this field is added to the semaphore count in an atomic way, such that the simultaneous software adds and DCP hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two. When the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore can be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

7.4.30 DCP channel 3 status register (DCP_CH3STAT)

The DCP channel 3 interrupt status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

DCP_CH3STAT: 0x1E0

CH3STAT_SET: 0x1E4

CH3STAT_CLR: 0x1E8

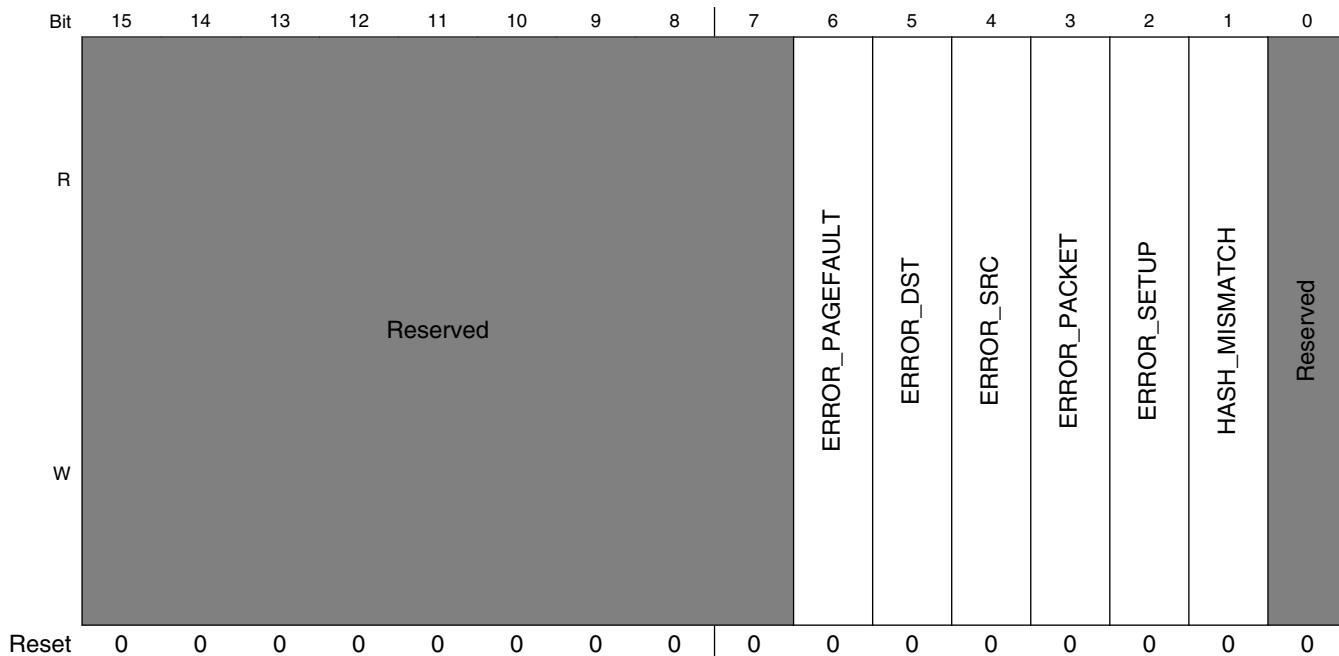
CH3STAT_TOG: 0x1EC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated once the packet is completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and the processing of the command chain is halted.

Address: 402F_C000h base + 1E0h offset = 402F_C1E0h

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

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

**DCP_CH3STAT field descriptions**

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure.
23–16 ERROR_CODE	Indicates additional error codes for some of the error conditions. 0x01 NEXT_CHAIN_IS_0 — Error is signalled because the next pointer is 0x00000000. 0x02 NO_CHAIN — Error is signalled because the semaphore is of a non-zero value and neither of the chain bits is set. 0x03 CONTEXT_ERROR — Error is signalled because an error was reported while reading/writing the context buffer. 0x04 PAYOUTLOAD_ERROR — Error is signalled because an error was reported while reading/writing the payload. 0x05 INVALID_MODE — Error is signalled because the control packet specifies an invalid mode select (for example, blit + hash).
15–7 -	This field is reserved. Reserved, always set to zero
6 ERROR_PAGEFAULT	This bit indicates that a page fault occurred while converting a virtual address to a physical address. When an error is detected, the channel's processing stops until the error is handled by the software.
5 ERROR_DST	This bit indicates that a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
4 ERROR_SRC	This bit indicates that a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by the software.
3 ERROR_PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing the status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by the software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration (such as a buffer length that is not a multiple of the natural data size for the operation). When an error is detected, the channel's processing stops until the error is handled by the software.

Table continues on the next page...

DCP_CH3STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	This bit indicates that a hashing check operation is mismatched for the control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by the software.
0 RSVD_ COMPLETE	This field is reserved. This bit always reads 0 in the status register, but is set to 1 in the packet status field after processing of the packet completes. This was done so that the software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

7.4.31 DCP channel 3 options register (DCP_CH3OPTS)

The DCP channel 3 options status register contains optional control information that may be used to further tune the behavior of the channel.

DCP CH3OPTS: 0x1F0

CH3OPTS SET: 0x1F4

CH3OPTS CLR: 0x1F8

CH3OPTS TOG: 0x1FC

The options register can be used to control the optional features of the channels.

Address: 402F_C000h base + 1F0h offset = 402F_C1F0h

DCP CH3OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero
RECOVERY_TIMER	This field indicates the recovery time for the channel. After each operation, the recovery timer for the channel is initialized with this value and then decremented until the timer reaches zero. The channel does not initiate the operation on the next packet in the chain until the recovery time expires. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range from 0 ns to 8.3 ms at the operation frequency of 133 MHz.

7.4.32 DCP debug select register (DCP_DBGSELECT)

This register selects a debug register to view.

DCP memory map/register definition

This register selects the debug information to return in the debug data register.

Address: 402F_C000h base + 400h offset = 402F_C400h

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

Reset 0

DCP_DBGSELECT field descriptions

Field	Description
31–8 RSVD	This field is reserved. Reserved, always set to zero
INDEX	Selects a value to read via the debug data register. 0x01 CONTROL — 0x10 OTPKEY0 — 0x11 OTPKEY1 — 0x12 OTPKEY2 — 0x13 OTPKEY3 —

7.4.33 DCP debug data register (DCP_DBGDATA)

Reading this register returns the debug data value from the selected index.

This register returns the debug data from the selected debug index source.

Address: 402F_C000h base + 410h offset = 402F_C410h

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

Reset 0

DCP_DBGDATA field descriptions

Field	Description
DATA	Debug data

7.4.34 DCP page table register (DCP_PAGETABLE)

The DCP page table register controls the virtual memory functionality of the DCP. It provides a base address for the page table as well as an enable/disable bit and the ability to flush the cached page table entries.

This register returns the debug data from the selected debug index source.

Address: 402F_C000h base + 420h offset = 402F_C420h

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

DCP_PAGETABLE field descriptions

Field	Description
31–2 BASE	Page table base address. The page table must be word-aligned and the pointer must reference a page table in the standard ARM format.
1 FLUSH	Page table flush control. To flush the TLB, write this bit to 1 and then back to 0.
0 ENABLE	Page table enable control. The virtual addressing is only used when this bit is set to 1. Disabling the page table does not flush any cached entries, so the software must write the FLUSH high and enable the LOW when updating the page tables.

7.4.35 DCP version register (DCP_VERSION)

Read-only register indicating the implemented version of the DCP.

This register returns the debug data from the selected debug index source.

Address: 402F_C000h base + 430h offset = 402F_C430h

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

DCP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR version of the design implementation.
23–16 MINOR	Fixed read-only value reflecting the MINOR version of the design implementation.
STEP	Fixed read-only value reflecting the stepping of the version of the design implementation.

Chapter 8

On-Chip OTP Controller (OCOTP_CTRL)

8.1 Overview

This section contains information describing the requirements for the on-chip eFuse OTP controller along with details about the block functionality and implementation.

In this document, the words "eFuse" and "OTP" are interchangeable. OCOTP refers to the hardware block itself.

8.1.1 Features

The OCOTP provides the following features :

- 32-bit word restricted program and read to of eFuse OTP.
- Loading and housing of fuse content into shadow registers.
- Memory-mapped (restricted) access to shadow registers.
- Generation of HWV_FUSE (hardware visible fuse bus) and the HWV_REG bus which is made of up of volatile PIO register based "fuses". The HWV_REG bits come from the SCS (Software Controllable Signals) register.
- Generation of STICKY_REG which consists of sticky register bits.
- Provides program-protect and read-protect eFuse.
- Provide override and read protection of shadow register.
- CRC32 test for read-lock fuse content.

8.2 Clocks

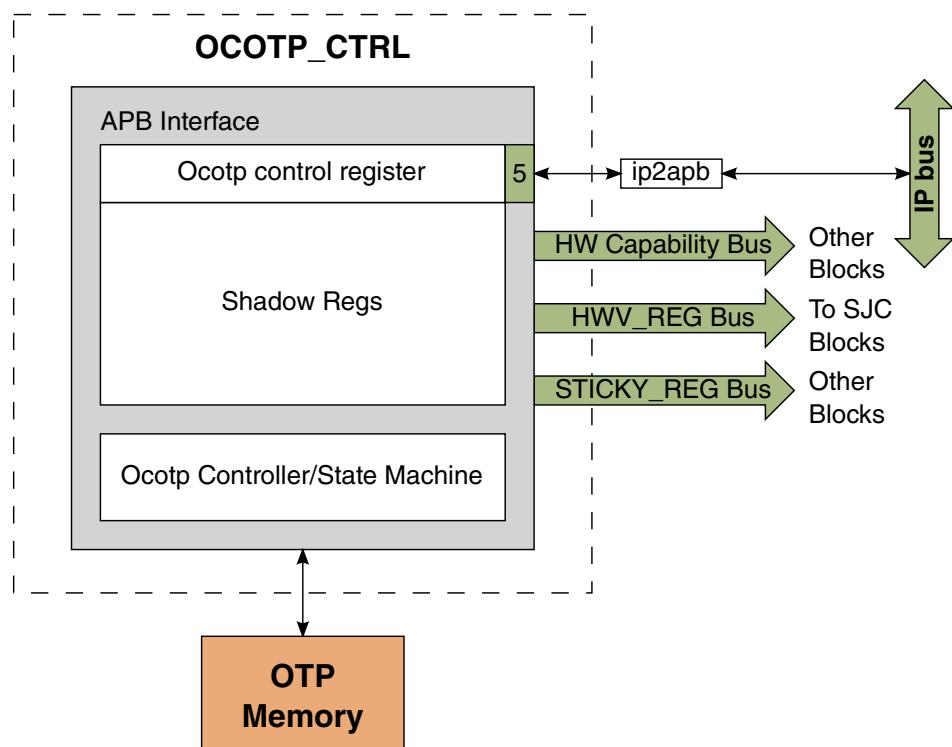
The table found here describes the clock sources for OCOTP. Please see the chip-specific clocking section for clock setting, configuration and gating information.

Table 8-1. OCOTP Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

8.3 Top-Level Symbol and Functional Overview

The figure below shows the OCOTP system level diagram.

**Figure 8-1. OCOTP System Level Diagram**

8.3.1 Operation

The IP bus interface of the OCOTP has the following functions.

- Configure control registers for programming and reading fuse wordbank.
- Override and read shadow registers.

OCOTP configuration for program and read are performed on 32-bit words for SW convenience. For writes, the 32-bit word reflects the "write-mask". Bit fields with 0 will not be programmed and bit fields with 1 will be programmed. OCOTP will program bit field with 1 in the fuse word one bit by one bit. For reads, OCOTP will read 4 times to get 4 bytes in the fuse word in order.

8.3.1.1 Shadow Register Reload

All fuse words in efusebox are shadowed. Therefore, fuse information is available through memory mapped shadow registers. If fuses are subsequently programmed, the shadow registers should be reloaded to keep them coherent with the fuse bank arrays.

The "reload shadows" feature allows the user to force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device. To force a reload, complete the following steps:

1. Set the HW_OCOTP_TIMING[STROBE_READ] field value appropriately.
2. Set the HW_OCOTP_TIMING[RELAX] field value appropriately.
3. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write , read or reload must be completed before a new access can be requested.
4. Set the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit. OCOTP will read all the fuses one by one and put it into corresponding shadow register.
5. Wait for HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[RELOAD_SHADOWS] to be cleared by the controller.

The controller will automatically clear the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

8.3.1.2 Fuse and Shadow register read

All shadow registers are always readable through the APB bus except some secret keys regions. When their corresponding fuse lock bits are set, the shadow registers also become read locked. After read locking, reading from these registers will return 0xBADABADA.

In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write , read or reload access can be issued. Subsequent reads to unlocked shadow locations will still work successfully however.

To read fuse words directly from fusebox correctly complete the following steps:

1. Program HW_OCOTP_TIMING[STROBE_READ] and HW_OCOTP_TIMING[RELAX] fields with timing values to match the current frequency of the ipg_clk. OTP read will work at maximum bus frequencies as long as the HW_OCOTP_TIMING parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read or reload must be completed before a read access can be requested.
3. Write the requested address to HW_OCOTP_CTRL[ADDR].
4. Set HW_OCOTP_READ_CTRL[READ_FUSE] to 1. OCOTP will auto read the fuse word according to HW_OCOTP_CTRL[ADDR] in fusebox. Then put read value into HW_OCOTP_READ_FUSE_DATA
5. Once complete, the controller will clear BUSY. A read request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new access can be issued.
6. Read HW_OCOTP_READ_FUSE_DATA to get fuse word value. HW_OCOTP_READ_FUSE_DATA will be 0xBADABADA when HW_OCOTP_CTRL[ERROR] is set.

8.3.1.3 Fuse and Shadow Register Writes

Shadow register bits can be overridden by software until the corresponding fuse lock bit for the region is set. When the lock shadow bit is set, the shadow registers for that lock region become write locked. The LOCK shadow register also has no shadow or fuse lock bits but it is always read only.

In order to avoid "rogue" code performing erroneous writes to OTP, a special unlocking sequence is required for writes to the fuse banks. To program fuse bank correctly complete the following steps:

1. Program the following fields with timing values to match the frequency of ipg_clk:
 - HW_OCOTP_TIMING[STROBE_PROG]
 - HW_OCOTP_TIMING[RELAX]
 OTP writes will work at maximum bus frequencies as long as the HW_OCOTP_TIMING parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write or reload must be completed before a write access can be requested.
3. Write the requested address to HW_OCOTP_CTRL[ADDR] and program the unlock code into HW_OCOTP_CTRL[WR_UNLOCK]. This must be programmed for each

write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation. .

4. Write the data to the HW_OCOTP_DATA register. This will automatically set HW_OCOTP_CTRL[BUSY] and clear HW_OCOTP_CTRL[WR_UNLOCK]. Bit fields with 1's will result in that OTP bit being programmed. Bit fields with 0's will be ignored. At the same time that the write is accepted, the controller makes an internal copy of HW_OCOTP_CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL[ADDR] will not affect an active write operation. During the write operation, HW_OCOTP_DATA cannot be modified.
5. Once complete, the controller will clear BUSY. A write request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are numbers of 10 micro-seconds per word. Write latencies is based on amount of bit filed which is 1. For example : program half fuse bits in one word need 10 us x 16.

For further details of OTP read/write operations see [eFUSE].

HW_OCOTP_CTRL[ERROR] will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (essentially, while HW_OCOTP_CTRL[RELOAD_SHADOWS] is set. In addition, the contents of the shadow register shall not be updated.
- A write is performed to a shadow register which has been locked.
- A read is performed to from a shadow register which has been read locked.
- A program is performed to a fuse word which has been locked.
- A read is performed to a fuse word which has been read locked.

8.3.1.4 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 us after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes.

A recommended software sequence to meet the postamble requirements is as follows:

- Issue the write and poll for BUSY (as per Fuse Shadow Memory Footprint).

- Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 us.
- Perform the next OTP operation.

8.3.2 Fuse Shadow Memory Footprint

The OTP memory footprint shows in the following figure. The registers are grouped by lock region. Their names correspond to the PIO register and fusemap names.

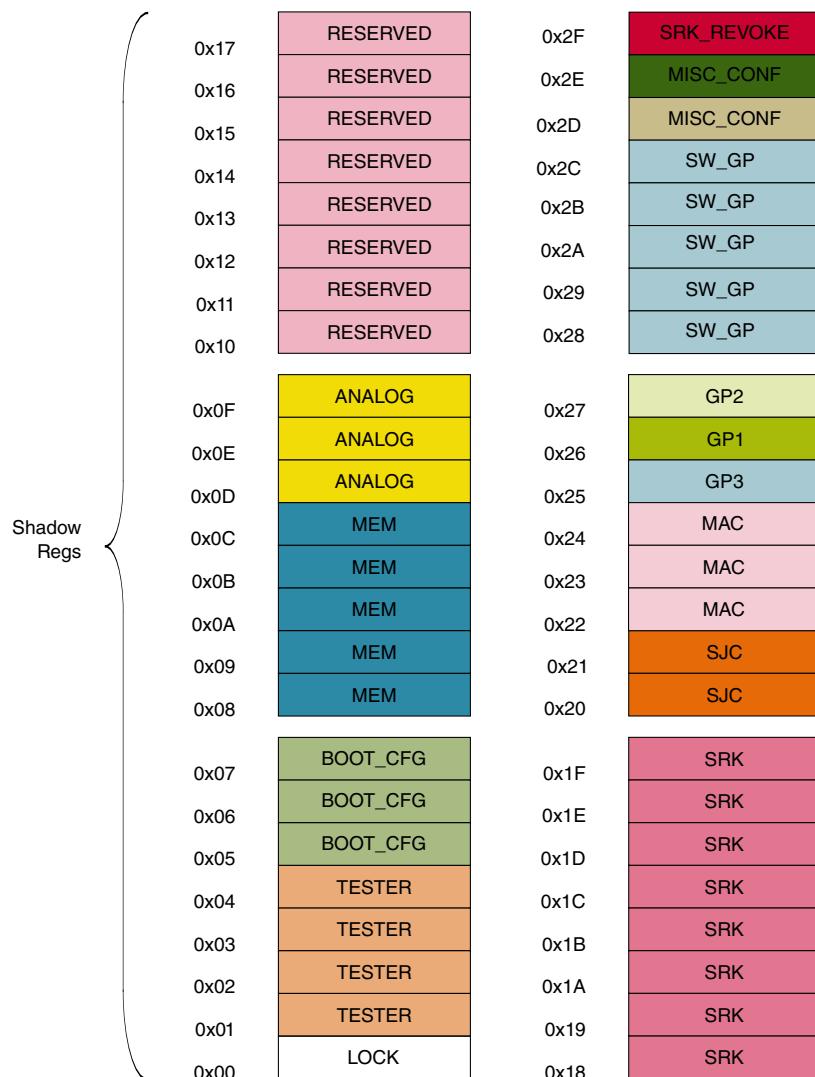


Figure 8-2. OTP Memory Footprint

8.3.3 OTP Read/Write Timing Parameters

The timing fields contained in the HW_OCOTP_TIMING register that specify counter limit values, which are used to time how long the state machine remains in the various states, as well as specify the STROBE signal timing.

The timing parameters are specified in ipg_clk cycles. Since the ipg_clk frequency can be set to a range of values, these parameters must be adjusted with the clock to yield the appropriate delay.

The HW_OCOTP_TIMING[RELAX] field specifies how long to remain in states to meet setup and hold timing requirement in fuse spec. This parameter should be set by the following equation:

$$t_{RELAX} = t_{HP_PG} = (HW_OCOTP_TIMING[RELAX]+1)/ipg_frequency > 16.2\text{ns}$$

HW_OCOTP_TIMING[RELAX] field is used to create other setup and hold timing delays in addition to tHP_PG. For all timing to be met, this is the max delay that must be programmed.

Except for setup and hold timing delay, there are 2 timing parameters for STROBE signal pulse width in program and read.

The HW_OCOTP_TIMING[STROBE_PROG] field specifies the period of the STROBE signal for fuse writes and is given in units of ipg_clk cycles. This value should be specified so that the requirement for the time when the STROBE signal is asserted high is met: $9000\text{ns} < t_{PGM} < 11000\text{ns}$ is met. Even though a range is given for tPGM, it is advised in [eFUSE] to program for a value of 10000ns. Therefore, this field should be set according to the following equation:

$$t_{PGM} = ((HW_OCOTP_TIMING[STROBE_PROG]+1) - 2*(HW_OCOTP_TIMING[RELAX]+1))/ipg_frequency = 10000\text{ns}.$$

The HW_OCOTP_TIMING[STROBE_READ] field specifies the period of the STROBE signal for fuse reads and is given in units of ipg_clk cycles. This field should be set according to the following equation:

$$t_{RD} = ((HW_OCOTP_TIMING[STROBE_READ]+1) - 2*(HW_OCOTP_TIMING[RELAX]+1))/ipg_frequency > 40 \text{ ns.}$$

The figure below illustrates the relationship between the STROBE signal in programming and reading mode, as well as the timing PIO register fields that affect it. The implementation uses one counter to generate the STROBE waveform within one period and a second counter counts the number of cycles to create for programming the designated word.

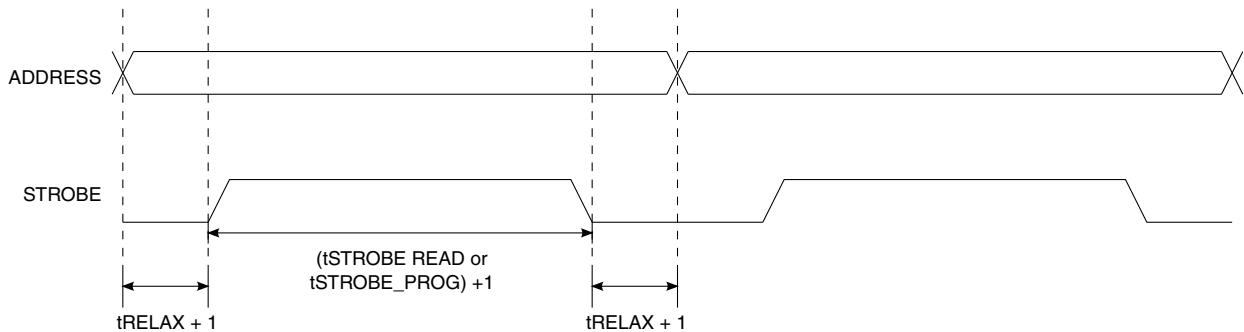


Figure 8-3. STROBE Signal Creation and Timing

8.3.4 Hardware Visible Fuses

The hwv_fuse bus emanates from the OCOTP block and goes to various other blocks inside the chip. This bus is made up of the shadow register bits .

Only a subset of these fuse bits are currently used by the hardware. The fuse bits are initially copied from the banks after reset is deasserted. When all fuse bits are loaded into their shadow registers, the OCOTP asserts the fuse_latched output signal.

The hwv_reg bus also comes from the OCOTP. Its source is the HW_OCOTP_SCS register. This register has 1 defined bit, the HAB_JDE bit, that is connected to the SJC block. The SCS bits are intended to be used as volatile fuse bits under software control. Additional bits will be defined as needed in future implementations.

The system-wide reset sequence must be coordinated by the system reset controller, so that the hwv_fuse and hwv_reg busses are stable and reflect the values of the fuses before they are used by the rest of the system.

8.3.5 Behavior During Reset

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is similar to that of a "reload shadow" operation.

8.3.6 Secure JTAG control

The JTAG control fuses are used to allow or disallow JTAG access to secured resources.

Three JTAG security levels are envisioned, as shown in the table below.

Table 8-2. JTAG Security Level Control Bits

Security Mode	JTAG_SMODE	Description
No Debug	2'b11	The highest security level.
Secure JTAG	2'b01	Limit the JTAG access by using key based authentication mechanism.
JTAG Enable	2'b00	Low Security, all JTAG features are enabled.

8.4 Fuse Map

See the Fusemap chapter of this reference manual for more information.

8.5 OCOTP Memory Map/Register Definition

The OCOTP Memory Map/Register Definition can be found here.

8.5.1 OCOTP register descriptions

8.5.1.1 OCOTP Memory map

OCOTP base address: 401F_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	OTP Controller Control Register (HW_OCOTP_CTRL)	32	RW	0000_0000h
4h	OTP Controller Control Register (HW_OCOTP_CTRL_SET)	32	RW	0000_0000h
8h	OTP Controller Control Register (HW_OCOTP_CTRL_CLR)	32	RW	0000_0000h
Ch	OTP Controller Control Register (HW_OCOTP_CTRL_TOG)	32	RW	0000_0000h
10h	OTP Controller Timing Register (HW_OCOTP_TIMING)	32	RW	060D_9755h
20h	OTP Controller Write Data Register (HW_OCOTP_DATA)	32	RW	0000_0000h
30h	OTP Controller Write Data Register (HW_OCOTP_READ_CTRL)	32	RW	0000_0000h
40h	OTP Controller Read Data Register (HW_OCOTP_READ_FUSE_D ATA)	32	RW	0000_0000h
50h	Sticky bit Register (HW_OCOTP_SW_STICKY)	32	RW	0000_0000h

Table continues on the next page...

OCOTP Memory Map/Register Definition

Offset	Register	Width (In bits)	Access	Reset value
60h	Software Controllable Signals Register (HW_OCOTP_SCS)	32	RW	0000_0000h
64h	Software Controllable Signals Register (HW_OCOTP_SCS_SET)	32	RW	0000_0000h
68h	Software Controllable Signals Register (HW_OCOTP_SCS_CLR)	32	RW	0000_0000h
6Ch	Software Controllable Signals Register (HW_OCOTP_SCS_TOG)	32	RW	0000_0000h
90h	OTP Controller Version Register (HW_OCOTP_VERSION)	32	RO	0600_0000h
100h	OTP Controller Timing Register 2 (HW_OCOTP_TIMING2)	32	RW	01C3_0092h
400h	Value of OTP Bank0 Word0 (Lock controls) (HW_OCOTP_LOCK)	32	RW	0000_0000h
410h	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG0)	32	RW	0000_0000h
420h	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG1)	32	RW	0000_0000h
430h	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG2)	32	RW	0000_0000h
440h	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG3)	32	RW	0000_0000h
450h	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG4)	32	RW	0000_0000h
460h	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG5)	32	RW	0000_0000h
470h	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG6)	32	RW	0000_0000h
480h	Value of OTP Bank1 Word0 (Memory Related Info.) (HW_OCOTP_MEM0)	32	RW	0000_0000h
490h	Value of OTP Bank1 Word1 (Memory Related Info.) (HW_OCOTP_MEM1)	32	RW	0000_0000h
4A0h	Value of OTP Bank1 Word2 (Memory Related Info.) (HW_OCOTP_MEM2)	32	RW	0000_0000h
4B0h	Value of OTP Bank1 Word3 (Memory Related Info.) (HW_OCOTP_MEM3)	32	RW	0000_0000h
4C0h	Value of OTP Bank1 Word4 (Memory Related Info.) (HW_OCOTP_MEM4)	32	RW	0000_0000h
4D0h	Value of OTP Bank1 Word5 (Analog Info.) (HW_OCOTP_ANA0)	32	RW	0000_0000h
4E0h	Value of OTP Bank1 Word6 (Analog Info.) (HW_OCOTP_ANA1)	32	RW	0000_0000h
4F0h	Value of OTP Bank1 Word7 (Analog Info.) (HW_OCOTP_ANA2)	32	RW	0000_0000h
580h	Shadow Register for OTP Bank3 Word0 (SRK Hash) (HW_OCOTP_SRK0)	32	RW	0000_0000h
590h	Shadow Register for OTP Bank3 Word1 (SRK Hash) (HW_OCOTP_SRK1)	32	RW	0000_0000h
5A0h	Shadow Register for OTP Bank3 Word2 (SRK Hash) (HW_OCOTP_SRK2)	32	RW	0000_0000h
5B0h	Shadow Register for OTP Bank3 Word3 (SRK Hash) (HW_OCOTP_SRK3)	32	RW	0000_0000h
5C0h	Shadow Register for OTP Bank3 Word4 (SRK Hash) (HW_OCOTP_SRK4)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
5D0h	Shadow Register for OTP Bank3 Word5 (SRK Hash) (HW_OCOTP_SRK5)	32	RW	0000_0000h
5E0h	Shadow Register for OTP Bank3 Word6 (SRK Hash) (HW_OCOTP_SRK6)	32	RW	0000_0000h
5F0h	Shadow Register for OTP Bank3 Word7 (SRK Hash) (HW_OCOTP_SRK7)	32	RW	0000_0000h
600h	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP0)	32	RW	0000_0000h
610h	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP1)	32	RW	0000_0000h
620h	Value of OTP Bank4 Word2 (MAC Address) (HW_OCOTP_MAC0)	32	RW	0000_0000h
630h	Value of OTP Bank4 Word3 (MAC Address) (HW_OCOTP_MAC1)	32	RW	0000_0000h
640h	Value of OTP Bank4 Word4 (MAC Address) (HW_OCOTP_GP3)	32	RW	0000_0000h
660h	Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (HW_OCOTP_GP1)	32	RW	0000_0000h
670h	Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (HW_OCOTP_GP2)	32	RW	0000_0000h
680h	Value of OTP Bank5 Word0 (SW GP1) (HW_OCOTP_SW_GP1)	32	RW	0000_0000h
690h	Value of OTP Bank5 Word1 (SW GP2) (HW_OCOTP_SW_GP20)	32	RW	0000_0000h
6A0h	Value of OTP Bank5 Word2 (SW GP2) (HW_OCOTP_SW_GP21)	32	RW	0000_0000h
6B0h	Value of OTP Bank5 Word3 (SW GP2) (HW_OCOTP_SW_GP22)	32	RW	0000_0000h
6C0h	Value of OTP Bank5 Word4 (SW GP2) (HW_OCOTP_SW_GP23)	32	RW	0000_0000h
6D0h	Value of OTP Bank5 Word5 (Misc Conf) (HW_OCOTP_MISC_CO_NF0)	32	RW	0000_0000h
6E0h	Value of OTP Bank5 Word6 (Misc Conf) (HW_OCOTP_MISC_CO_NF1)	32	RW	0000_0000h
6F0h	Value of OTP Bank5 Word7 (SRK Revoke) (HW_OCOTP_SRK_REV_OKE)	32	RW	0000_0000h

8.5.1.2 OTP Controller Control Register (HW_OCOTP_CTRL)

8.5.1.2.1 Offset

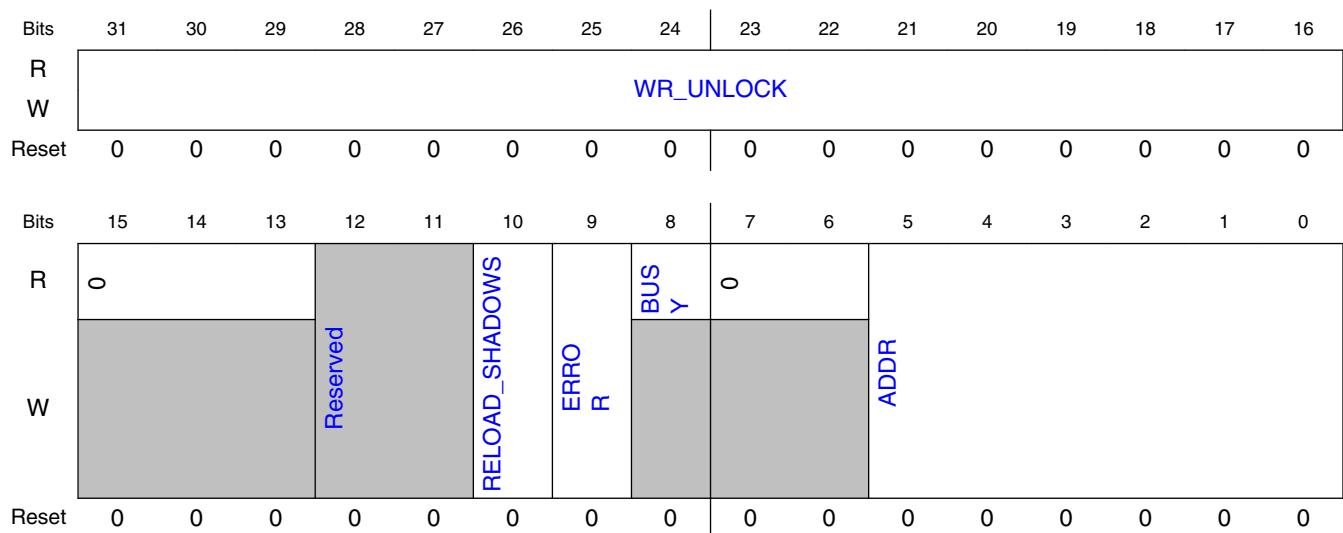
Register	Offset
HW_OCOTP_CTRL	0h

8.5.1.2.2 Function

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable

ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjunction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

8.5.1.2.3 Diagram



8.5.1.2.4 Fields

Field	Function
31-16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or

Table continues on the next page...

Field	Function
BUSY	after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

8.5.1.3 OTP Controller Control Register (HW_OCOTP_CTRL_SET)

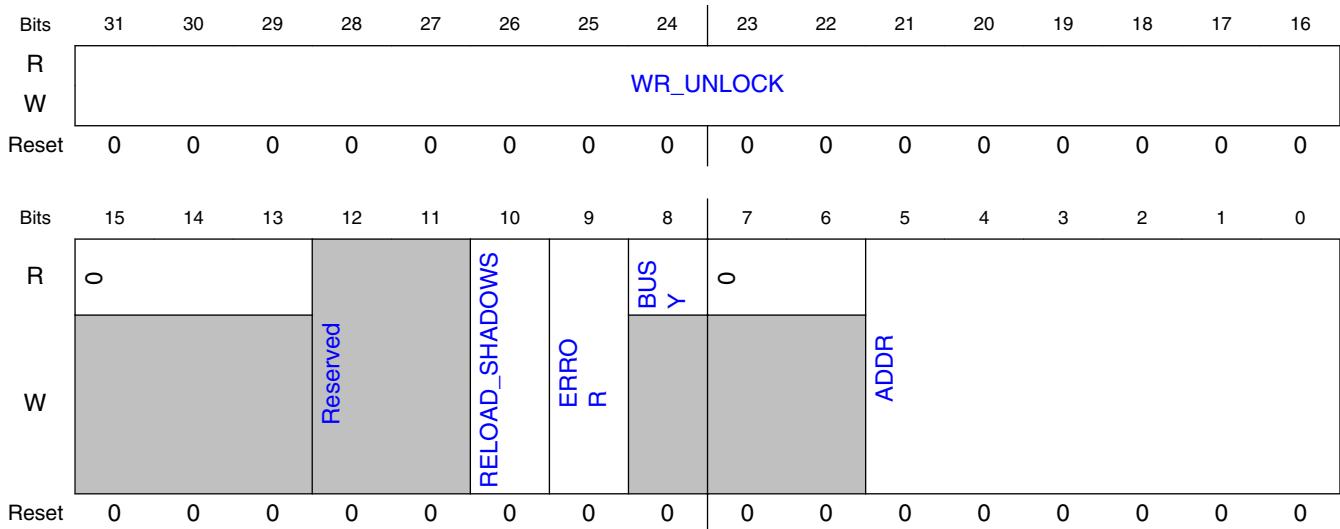
8.5.1.3.1 Offset

Register	Offset
HW_OCOTP_CTRL_SET	4h

8.5.1.3.2 Function

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjunction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

8.5.1.3.3 Diagram



8.5.1.3.4 Fields

Field	Function
31-16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

8.5.1.4 OTP Controller Control Register (HW_OCOTP_CTRL_CLR)

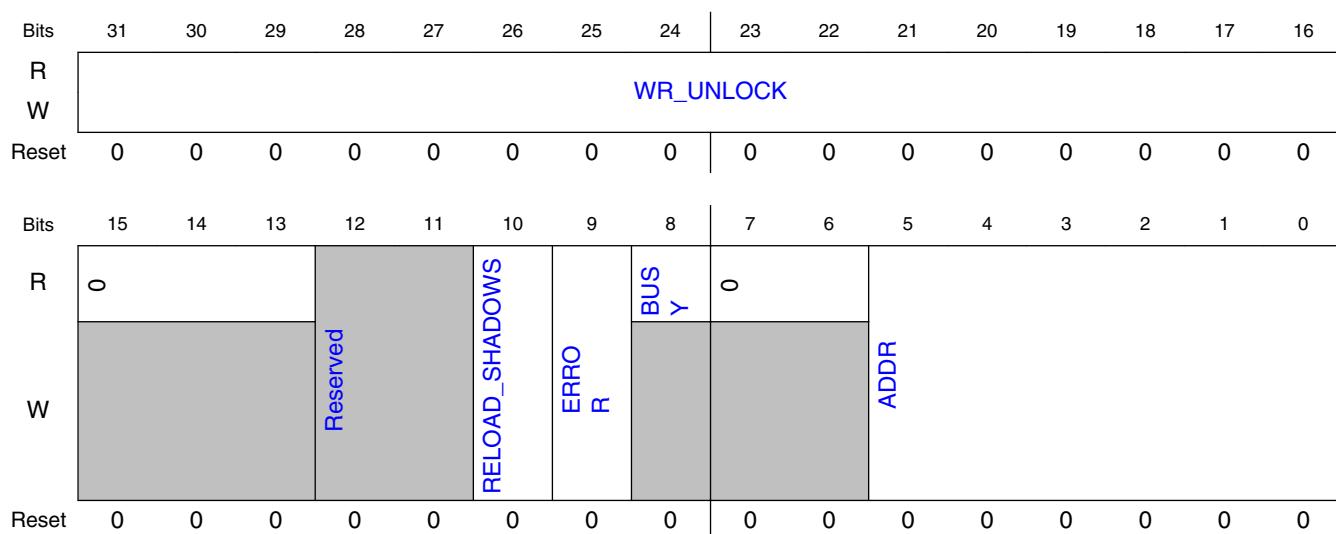
8.5.1.4.1 Offset

Register	Offset
HW_OCOTP_CTRL_CL R	8h

8.5.1.4.2 Function

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjunction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

8.5.1.4.3 Diagram



8.5.1.4.4 Fields

Field	Function
31-16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHAD_OWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

8.5.1.5 OTP Controller Control Register (HW_OCOTP_CTRL_TOG)

8.5.1.5.1 Offset

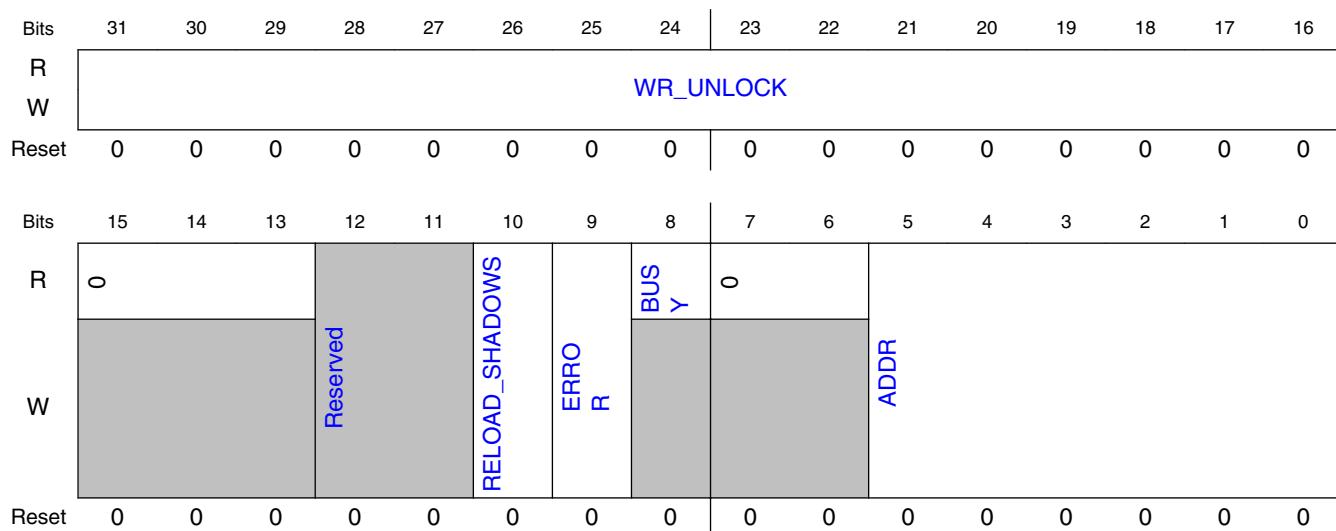
Register	Offset
HW_OCOTP_CTRL_TOG	Ch

8.5.1.5.2 Function

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjunction with the HW_OCOTP_DATA register to perform write operations.

Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

8.5.1.5.3 Diagram



8.5.1.5.4 Fields

Field	Function
31-16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.

Table continues on the next page...

OCOTP Memory Map/Register Definition

Field	Function
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

8.5.1.6 OTP Controller Timing Register (HW_OCOTP_TIMING)

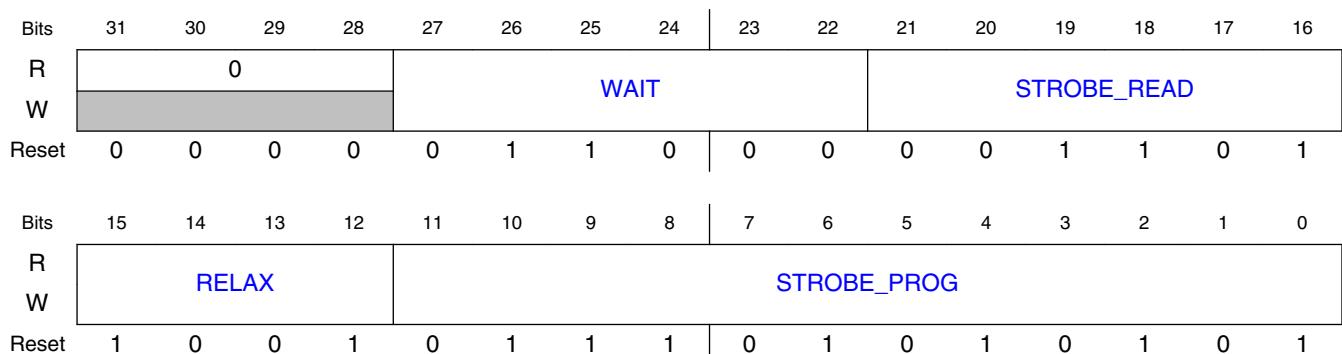
8.5.1.6.1 Offset

Register	Offset
HW_OCOTP_TIMING	10h

8.5.1.6.2 Function

This register specifies timing parameters for programming and reading the OCOTP fuse array.

8.5.1.6.3 Diagram



8.5.1.6.4 Fields

Field	Function
31-28 —	Reserved
—	

Table continues on the next page...

Field	Function
27-22 WAIT	This count value specifies time interval between auto read and write access in one time program. It is given in number of ipg_clk periods.
21-16 STROBE_READ	This count value specifies the strobe period in one time read OTP. $Trd = ((STROBE_READ+1)-2*(RELAX+1)) / \text{ipg_clk_freq}$. It is given in number of ipg_clk periods.
15-12 RELAX	This count value specifies the time to add to all default timing parameters other than the Tpgm and Trd. It is given in number of ipg_clk periods.
11-0 STROBE_PROG	This count value specifies the strobe period in one time write OTP. $Tpgm = ((STROBE_PROG+1)-2*(RELAX+1)) / \text{ipg_clk_freq}$. It is given in number of ipg_clk periods.

8.5.1.7 OTP Controller Write Data Register (HW_OCOTP_DATA)

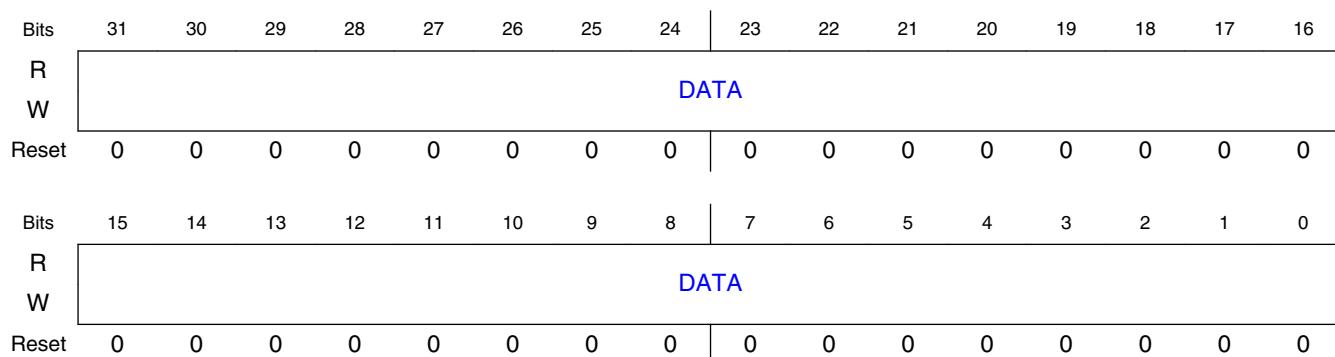
8.5.1.7.1 Offset

Register	Offset
HW_OCOTP_DATA	20h

8.5.1.7.2 Function

This register is used in conjunction with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

8.5.1.7.3 Diagram



8.5.1.7.4 Fields

Field	Function
31-0 DATA	Used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

8.5.1.8 OTP Controller Write Data Register (HW_OCOTP_READ_CTL)

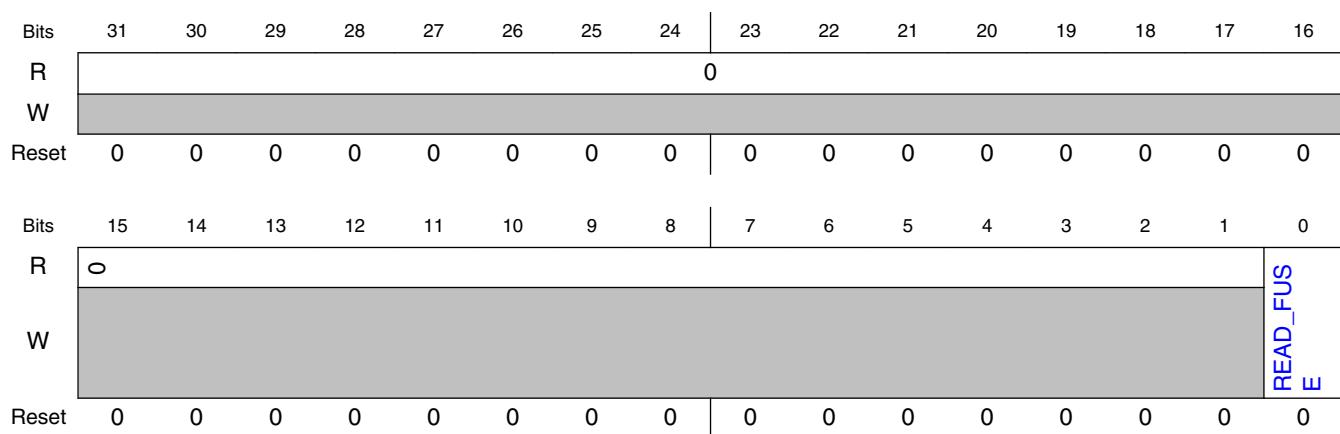
8.5.1.8.1 Offset

Register	Offset
HW_OCOTP_READ_CTL	30h

8.5.1.8.2 Function

This register is used in conjunction with HW_OCOTP_CTRL to perform one time read to the OTP. Please see the "Software read Sequence" section for operating details.

8.5.1.8.3 Diagram



8.5.1.8.4 Fields

Field	Function
31-1	Reserved

Table continues on the next page...

Field	Function
—	
0 READ_FUSE	Used to initiate a read to OTP. Please see the "Software read Sequence" section for operating details.

8.5.1.9 OTP Controller Read Data Register (HW_OCOTP_READ_FUSE_DATA)

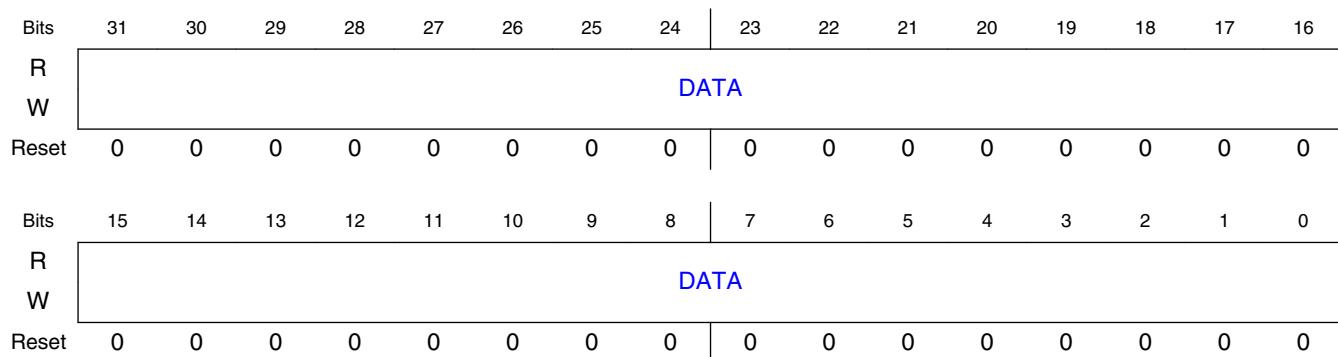
8.5.1.9.1 Offset

Register	Offset
HW_OCOTP_READ_FUSE_DATA	40h

8.5.1.9.2 Function

The data read from OTP

8.5.1.9.3 Diagram



8.5.1.9.4 Fields

Field	Function
31-0 DATA	The data read from OTP

8.5.1.10 Sticky bit Register (HW_OCOTP_SW_STICKY)

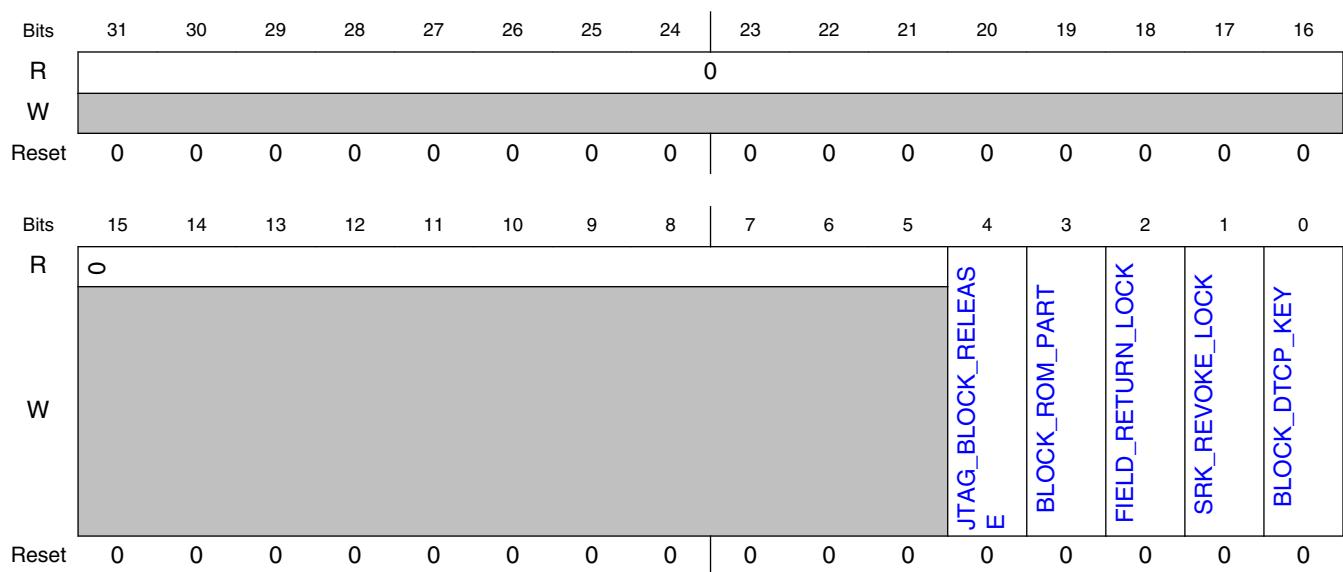
8.5.1.10.1 Offset

Register	Offset
HW_OCOTP_SW_STICKY	50h

8.5.1.10.2 Function

Some sticky bits are used by SW to lock some fuse area , shadow registers and other features.

8.5.1.10.3 Diagram



8.5.1.10.4 Fields

Field	Function
31-5 —	Reserved
4 JTAG_BLOCK_RELEASE	Set by ARM during Boot after DTCP is initialized and before test mode entry. * 0 (Default) - JTAG is blocked (subject to other conditions). * 1 - JTAG block is released (subject to other controls). Once this bit is set, it is always high unless a POR is issued.

Table continues on the next page...

Field	Function
3 BLOCK_ROM_PART	Set by ARM during Boot after DTCP is initialized and before test mode entry, if ROM_PART_LOCK=1.* 0 (Default) - Secret part of Boot ROM is not hidden (subject to other conditions). * 1 - Secret part of Boot ROM is hidden. Once this bit is set, it is always high unless a POR is issued.
2 FIELD_RETURN_LOCK	Shadow register write and OTP write lock for FIELD_RETURN region. When set, the writing of this region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
1 SRK_REVOKE_LOCK	Shadow register write and OTP write lock for SRK_REVOKER region. When set, the writing of this region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
0 BLOCK_DTCP_KEY	Shadow register read and OTP read lock for DTCP_KEY region. It is used to lock shadow register and OTP read for DTCP_KEY in conjunction with dtcp_disable and HDCP_KEY lock fuse bit. Once this bit is set, it is always high unless a POR is issued.

8.5.1.11 Software Controllable Signals Register (HW_OCOTP_SCS)

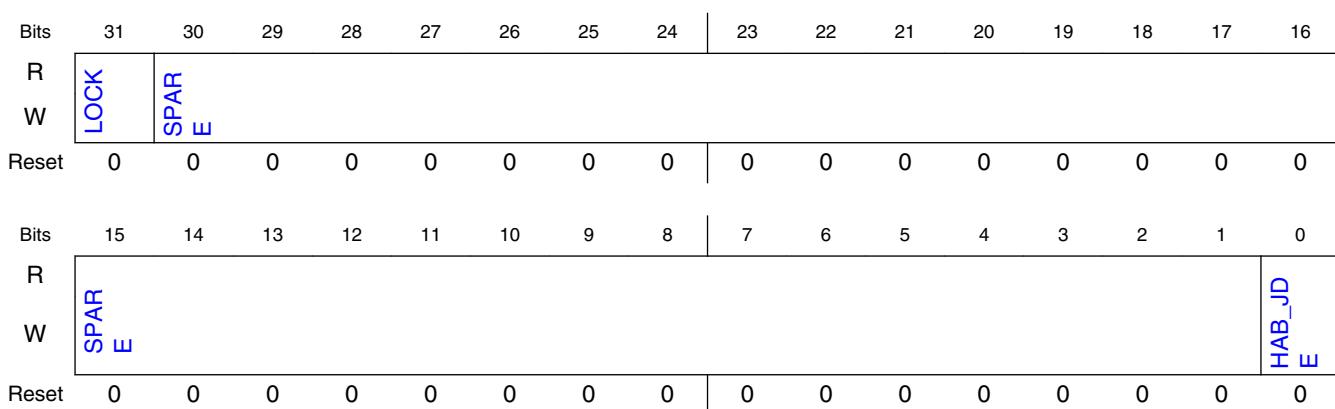
8.5.1.11.1 Offset

Register	Offset
HW_OCOTP_SCS	60h

8.5.1.11.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their default values after POR.

8.5.1.11.3 Diagram



8.5.1.11.4 Fields

Field	Function
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

8.5.1.12 Software Controllable Signals Register (HW_OCOTP_SCS_SET)

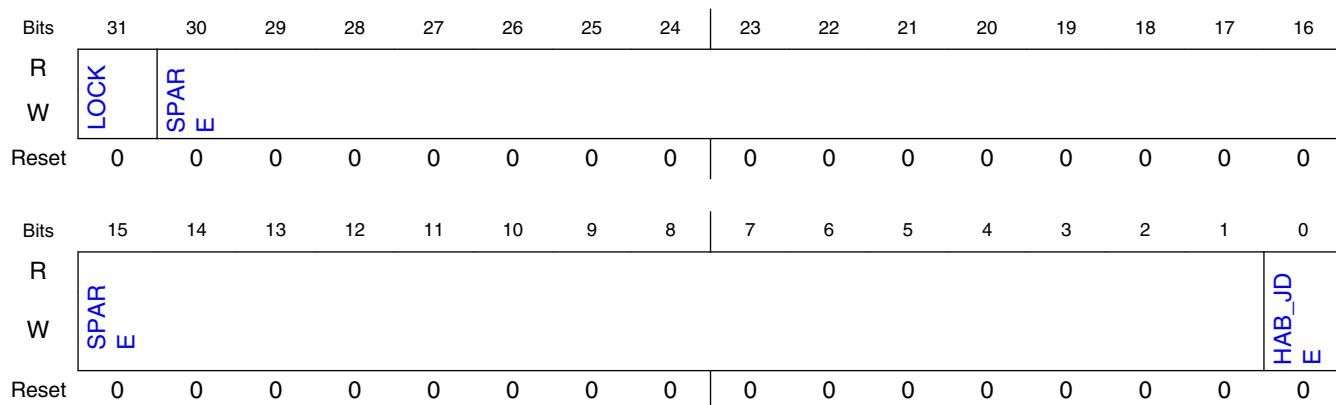
8.5.1.12.1 Offset

Register	Offset
HW_OCOTP_SCS_SET	64h

8.5.1.12.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

8.5.1.12.3 Diagram



8.5.1.12.4 Fields

Field	Function
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

8.5.1.13 Software Controllable Signals Register (HW_OCOTP_SCS_CLR)

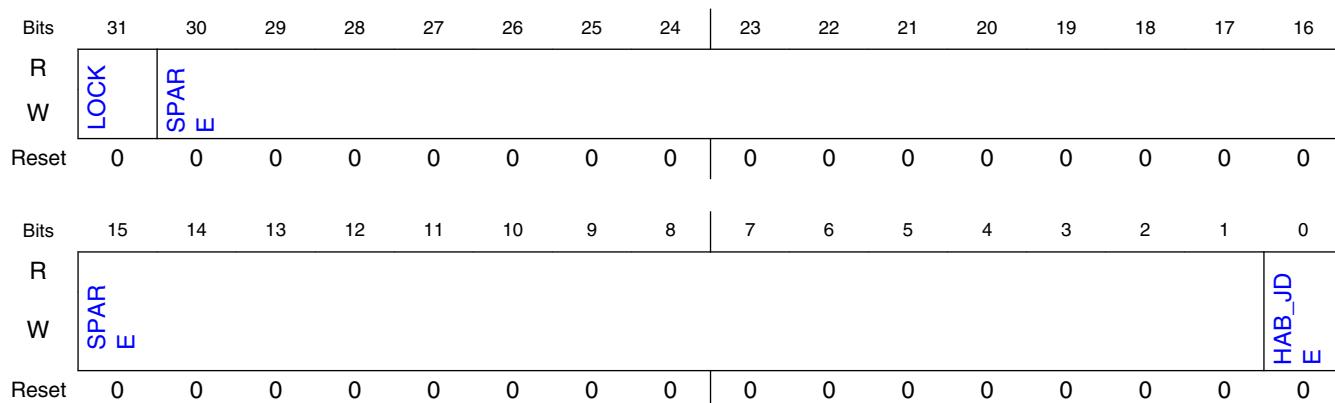
8.5.1.13.1 Offset

Register	Offset
HW_OCOTP_SCS_CLR	68h

8.5.1.13.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

8.5.1.13.3 Diagram



8.5.1.13.4 Fields

Field	Function
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

8.5.1.14 Software Controllable Signals Register (HW_OCOTP_SCS_TOG)

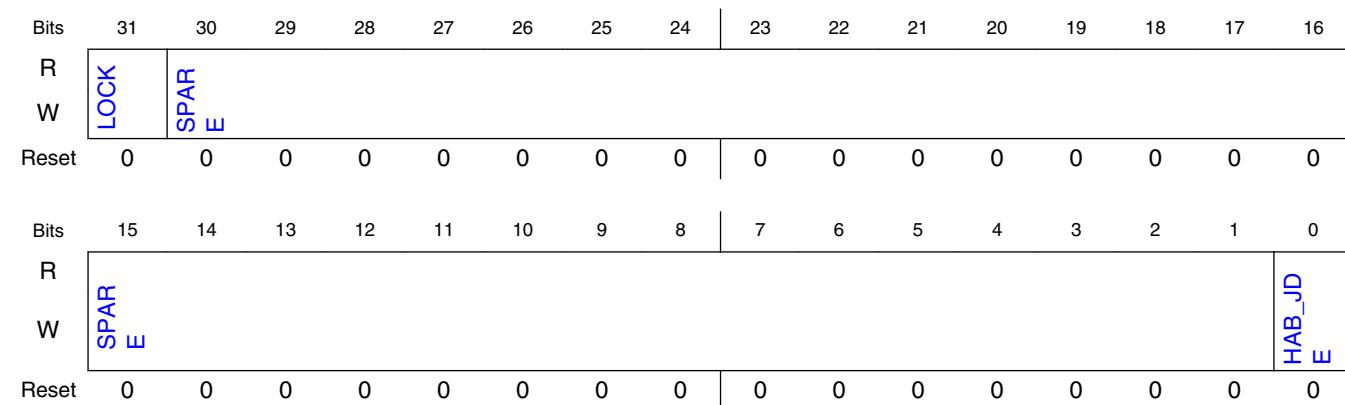
8.5.1.14.1 Offset

Register	Offset
HW_OCOTP_SCS_TOG	6Ch

8.5.1.14.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

8.5.1.14.3 Diagram



8.5.1.14.4 Fields

Field	Function
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

8.5.1.15 OTP Controller Version Register (HW_OCOTP_VERSION)

8.5.1.15.1 Offset

Register	Offset
HW_OCOTP_VERSION	90h

8.5.1.15.2 Function

This register indicates the RTL version in use.

8.5.1.15.3 Diagram

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

8.5.1.15.4 Fields

Field	Function
31-24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23-16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15-0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

8.5.1.16 OTP Controller Timing Register 2 (HW_OCOTP_TIMING2)

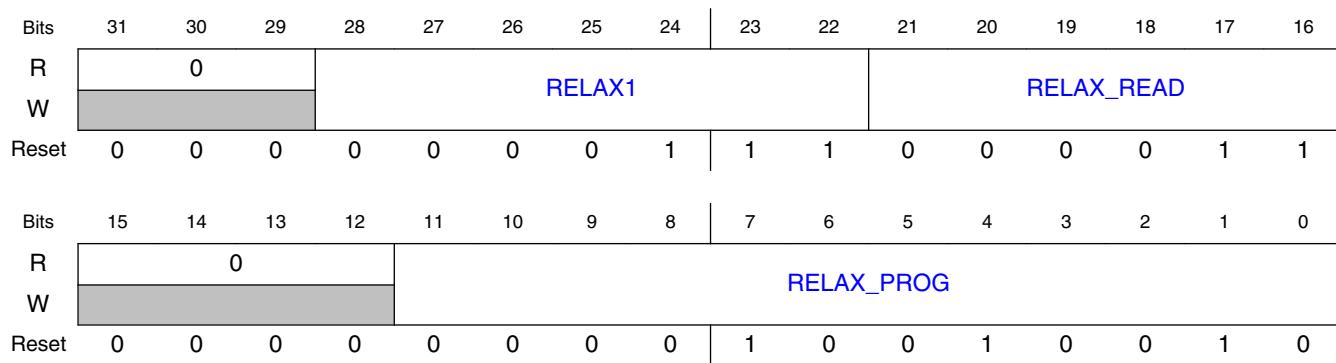
8.5.1.16.1 Offset

Register	Offset
HW_OCOTP_TIMING2	100h

8.5.1.16.2 Function

This register specifies timing parameters for programming and reading the OCOTP fuse array.

8.5.1.16.3 Diagram



8.5.1.16.4 Fields

Field	Function
31-29 —	Reserved
28-22 RELAX1	This count value specifies time interval between auto read and write access in one time program. It is given in number of ipg_clk periods.
21-16 RELAX_READ	This count value specifies the strobe period in one time read OTP. $T_{rd} = ((STROBE_READ+1) - 2*(RELAX_READ+1)) / \text{ipg_clk_freq}$. It is given in number of ipg_clk periods.
15-12 —	Reserved
11-0 RELAX_PROG	This count value specifies the strobe period in one time write OTP. $T_{pgm} = ((STROBE_PROG+1) - 2*(RELAX_PROG+1)) / \text{ipg_clk_freq}$. It is given in number of ipg_clk periods.

8.5.1.17 Value of OTP Bank0 Word0 (Lock controls) (HW_OCOTP_LOCK)

8.5.1.17.1 Offset

Register	Offset
HW_OCOTP_LOCK	400h

8.5.1.17.2 Function

Shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

8.5.1.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FIELD_RETUR N				GP3		0		SW_GP2_RLOCK	MISC_CONF	SW_GP2_LOCK	OTPMK_CRC	ANALOG		OTPMK LSB	SW_GP1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OTPMK_MSB	SR_K	GP2		GP1		MAC_ADDR		0	SJC_RES_P	MEM_TRIM		BOOT_CFG		TESTE_R	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

8.5.1.17.4 Fields

Field	Function
31-28 FIELD_RETUR N	Reserved
27-26 GP3	Status of shadow register and OTP write lock for gp3 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
25-24 —	Reserved
23 SW_GP2_RLO CK	Status of shadow register and OTP read lock for sw_gp2 region. When set, the reading of this region's shadow register and OTP fuse word are blocked.
22 MISC_CONF	Status of shadow register and OTP write lock for misc_conf region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
21 SW_GP2_LOCK	Status of shadow register and OTP write lock for sw_gp2 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
20 OTPMK_CRC	Status of shadow register and OTP write lock for otpmk_crc region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
19-18 ANALOG	Status of shadow register and OTP write lock for analog region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.

Table continues on the next page...

Field	Function
17 OTPMK_LSB	Status of shadow register read and write, OTP read and write lock for otpmk region (LSB). When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
16 SW_GP1	Status of shadow register and OTP write lock for sw_gp1 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
15 OTPMK_MSB	Status of shadow register read and write, OTP read and write lock for otpmk region (MSB). When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
14 SRK	Status of shadow register and OTP write lock for srk region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
13-12 GP2	Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
11-10 GP1	Status of shadow register and OTP write lock for gp1 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
9-8 MAC_ADDR	Status of shadow register and OTP write lock for mac_addr region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
7 —	Reserved
6 SJC_RESP	Status of shadow register read and write, OTP read and write lock for sjc_resp region. When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
5-4 MEM_TRIM	Status of shadow register and OTP write lock for mem_trim region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
3-2 BOOT_CFG	Status of shadow register and OTP write lock for boot_cfg region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
1-0 TESTER	Status of shadow register and OTP write lock for tester region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.

8.5.1.18 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG0)

8.5.1.18.1 Offset

Register	Offset
HW_OCOTP_CFG0	410h

8.5.1.18.2 Function

Shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

8.5.1.18.3 Diagram

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

8.5.1.18.4 Fields

Field	Function
31-0 BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 1 (ADDR = 0x01). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

8.5.1.19 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG1)

8.5.1.19.1 Offset

Register	Offset
HW_OCOTP_CFG1	420h

8.5.1.19.2 Function

shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

8.5.1.19.3 Diagram

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

8.5.1.19.4 Fields

Field	Function
31-0 BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 2 (ADDR = 0x02). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

8.5.1.20 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG2)

8.5.1.20.1 Offset

Register	Offset
HW_OCOTP_CFG2	430h

8.5.1.20.2 Function

Shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

8.5.1.20.3 Diagram

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

8.5.1.20.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

8.5.1.21 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG3)

8.5.1.21.1 Offset

Register	Offset
HW_OCOTP_CFG3	440h

8.5.1.21.2 Function

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

8.5.1.21.3 Diagram

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

8.5.1.21.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

8.5.1.22 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG4)

8.5.1.22.1 Offset

Register	Offset
HW_OCOTP_CFG4	450h

8.5.1.22.2 Function

Shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

8.5.1.22.3 Diagram

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

8.5.1.22.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

8.5.1.23 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG5)

8.5.1.23.1 Offset

Register	Offset
HW_OCOTP_CFG5	460h

8.5.1.23.2 Function

Shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

8.5.1.23.3 Diagram

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

8.5.1.23.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

8.5.1.24 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG6)

8.5.1.24.1 Offset

Register	Offset
HW_OCOTP_CFG6	470h

8.5.1.24.2 Function

Shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

8.5.1.24.3 Diagram

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

8.5.1.24.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

8.5.1.25 Value of OTP Bank1 Word0 (Memory Related Info.) (HW_OCOTP_MEM0)

8.5.1.25.1 Offset

Register	Offset
HW_OCOTP_MEM0	480h

8.5.1.25.2 Function

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

8.5.1.25.3 Diagram

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

8.5.1.25.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 0 (ADDR = 0x08). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

8.5.1.26 Value of OTP Bank1 Word1 (Memory Related Info.) (HW_OCOTP_MEM1)

8.5.1.26.1 Offset

Register	Offset
HW_OCOTP_MEM1	490h

8.5.1.26.2 Function

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

8.5.1.26.3 Diagram

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

8.5.1.26.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 1 (ADDR = 0x09). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

8.5.1.27 Value of OTP Bank1 Word2 (Memory Related Info.) (HW_OCOTP_MEM2)

8.5.1.27.1 Offset

Register	Offset
HW_OCOTP_MEM2	4A0h

8.5.1.27.2 Function

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

8.5.1.27.3 Diagram

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

8.5.1.27.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 2 (ADDR = 0x0A). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

8.5.1.28 Value of OTP Bank1 Word3 (Memory Related Info.) (HW_OCOTP_MEM3)

8.5.1.28.1 Offset

Register	Offset
HW_OCOTP_MEM3	4B0h

8.5.1.28.2 Function

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

8.5.1.28.3 Diagram

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

8.5.1.28.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 3 (ADDR = 0x0B). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

8.5.1.29 Value of OTP Bank1 Word4 (Memory Related Info.) (HW_OCOTP_MEM4)

8.5.1.29.1 Offset

Register	Offset
HW_OCOTP_MEM4	4C0h

8.5.1.29.2 Function

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

8.5.1.29.3 Diagram

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

8.5.1.29.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 4 (ADDR = 0x0C). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

8.5.1.30 Value of OTP Bank1 Word5 (Analog Info.) (HW_OCOTP_ANA0)

8.5.1.30.1 Offset

Register	Offset
HW_OCOTP_ANA0	4D0h

8.5.1.30.2 Function

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

8.5.1.30.3 Diagram

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

8.5.1.30.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 5 (ADDR = 0x0D). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

8.5.1.31 Value of OTP Bank1 Word6 (Analog Info.) (HW_OCOTP_ANA1)

8.5.1.31.1 Offset

Register	Offset
HW_OCOTP_ANA1	4E0h

8.5.1.31.2 Function

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

8.5.1.31.3 Diagram

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

8.5.1.31.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 6 (ADDR = 0x0E). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

8.5.1.32 Value of OTP Bank1 Word7 (Analog Info.) (HW_OCOTP_ANA2)

8.5.1.32.1 Offset

Register	Offset
HW_OCOTP_ANA2	4F0h

8.5.1.32.2 Function

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

8.5.1.32.3 Diagram

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

8.5.1.32.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP bank 1, word 7 (ADDR = 0x0F). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

8.5.1.33 Shadow Register for OTP Bank3 Word0 (SRK Hash) (HW_OCOTP_SRK0)

8.5.1.33.1 Offset

Register	Offset
HW_OCOTP_SRK0	580h

8.5.1.33.2 Function

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

8.5.1.33.3 Diagram

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

8.5.1.33.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word0 (Copy of OTP Bank 3, word 0 (ADDR = 0x1C)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.34 Shadow Register for OTP Bank3 Word1 (SRK Hash) (HW_OCOTP_SRK1)

8.5.1.34.1 Offset

Register	Offset
HW_OCOTP_SRK1	590h

8.5.1.34.2 Function

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

8.5.1.34.3 Diagram

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

8.5.1.34.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word1 (Copy of OTP Bank 3, word 1 (ADDR = 0x1D)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.35 Shadow Register for OTP Bank3 Word2 (SRK Hash) (HW_OCOTP_SRK2)

8.5.1.35.1 Offset

Register	Offset
HW_OCOTP_SRK2	5A0h

8.5.1.35.2 Function

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

8.5.1.35.3 Diagram

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

8.5.1.35.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word2 (Copy of OTP Bank 3, word 2 (ADDR = 0x1E)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.36 Shadow Register for OTP Bank3 Word3 (SRK Hash) (HW_OCOTP_SRK3)

8.5.1.36.1 Offset

Register	Offset
HW_OCOTP_SRK3	5B0h

8.5.1.36.2 Function

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

8.5.1.36.3 Diagram

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

8.5.1.36.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word3 (Copy of OTP Bank 3, word 3 (ADDR = 0x1F)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.37 Shadow Register for OTP Bank3 Word4 (SRK Hash) (HW_OCOTP_SRK4)

8.5.1.37.1 Offset

Register	Offset
HW_OCOTP_SRK4	5C0h

8.5.1.37.2 Function

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

8.5.1.37.3 Diagram

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

8.5.1.37.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word4 (Copy of OTP Bank 3, word 4 (ADDR = 0x20)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.38 Shadow Register for OTP Bank3 Word5 (SRK Hash) (HW_OCOTP_SRK5)

8.5.1.38.1 Offset

Register	Offset
HW_OCOTP_SRK5	5D0h

8.5.1.38.2 Function

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

8.5.1.38.3 Diagram

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

8.5.1.38.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x21)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.39 Shadow Register for OTP Bank3 Word6 (SRK Hash) (HW_OCOTP_SRK6)

8.5.1.39.1 Offset

Register	Offset
HW_OCOTP_SRK6	5E0h

8.5.1.39.2 Function

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

8.5.1.39.3 Diagram

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

8.5.1.39.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x22)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.40 Shadow Register for OTP Bank3 Word7 (SRK Hash) (HW_OCOTP_SRK7)

8.5.1.40.1 Offset

Register	Offset
HW_OCOTP_SRK7	5F0h

8.5.1.40.2 Function

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

8.5.1.40.3 Diagram

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

8.5.1.40.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word7 (Copy of OTP Bank 3, word 7 (ADDR = 0x23)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

8.5.1.41 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP0)

8.5.1.41.1 Offset

Register	Offset
HW_OCOTP_SJC_RES_P0	600h

8.5.1.41.2 Function

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

8.5.1.41.3 Diagram

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

8.5.1.41.4 Fields

Field	Function
31-0 BITS	Shadow register for the SJC_RESP Key word0 (Copy of OTP Bank 4, word 0 (ADDR = 0x20)). These bits can be not read and written after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

8.5.1.42 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP1)

8.5.1.42.1 Offset

Register	Offset
HW_OCOTP_SJC_RES P1	610h

8.5.1.42.2 Function

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

8.5.1.42.3 Diagram

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

8.5.1.42.4 Fields

Field	Function
31-0 BITS	Shadow register for the SJC_RESP Key word1 (Copy of OTP Bank 4, word 1 (ADDR = 0x21)). These bits can be not read and written after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

8.5.1.43 Value of OTP Bank4 Word2 (MAC Address) (HW_OCOTP_MAC0)

8.5.1.43.1 Offset

Register	Offset
HW_OCOTP_MAC0	620h

8.5.1.43.2 Function

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

8.5.1.43.3 Diagram

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

8.5.1.43.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 4, word 2 (ADDR = 0x22).
BITS	

8.5.1.44 Value of OTP Bank4 Word3 (MAC Address) (HW_OCOTP_MAC1)

8.5.1.44.1 Offset

Register	Offset
HW_OCOTP_MAC1	630h

8.5.1.44.2 Function

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

8.5.1.44.3 Diagram

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

8.5.1.44.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 4, word 3 (ADDR = 0x23).
BITS	

8.5.1.45 Value of OTP Bank4 Word4 (MAC Address) (HW_OCOTP_GP3)

8.5.1.45.1 Offset

Register	Offset
HW_OCOTP_GP3	640h

8.5.1.45.2 Function

Shadowed memory mapped access to OTP Bank 4, word 4 (ADDR = 0x24).

8.5.1.45.3 Diagram

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

8.5.1.45.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 4, word 4 (ADDR = 0x24).
BITS	

8.5.1.46 Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (HW_OCOTP_GP1)

8.5.1.46.1 Offset

Register	Offset
HW_OCOTP_GP1	660h

8.5.1.46.2 Function

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

8.5.1.46.3 Diagram

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

8.5.1.46.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 4, word 6 (ADDR = 0x26).
BITS	

8.5.1.47 Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (HW_OCOTP_GP2)

8.5.1.47.1 Offset

Register	Offset
HW_OCOTP_GP2	670h

8.5.1.47.2 Function

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

8.5.1.47.3 Diagram

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

8.5.1.47.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 4, word 7 (ADDR = 0x27).
BITS	

8.5.1.48 Value of OTP Bank5 Word0 (SW GP1) (HW_OCOTP_SW_GP1)

8.5.1.48.1 Offset

Register	Offset
HW_OCOTP_SW_GP1	680h

8.5.1.48.2 Function

Shadowed memory mapped access to OTP Bank 5, word 0 (ADDR = 0x28).

8.5.1.48.3 Diagram

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

8.5.1.48.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 0 (ADDR = 0x28).
BITS	

8.5.1.49 Value of OTP Bank5 Word1 (SW GP2) (HW_OCOTP_SW_GP20)

8.5.1.49.1 Offset

Register	Offset
HW_OCOTP_SW_GP20	690h

8.5.1.49.2 Function

Shadowed memory mapped access to OTP Bank 5, word 1 (ADDR = 0x29).

8.5.1.49.3 Diagram

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

8.5.1.49.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 1 (ADDR = 0x29).
BITS	

8.5.1.50 Value of OTP Bank5 Word2 (SW GP2) (HW_OCOTP_SW_GP21)

8.5.1.50.1 Offset

Register	Offset
HW_OCOTP_SW_GP21	6A0h

8.5.1.50.2 Function

Shadowed memory mapped access to OTP Bank 5, word 2 (ADDR = 0x2a).

8.5.1.50.3 Diagram

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

8.5.1.50.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 2 (ADDR = 0x2a).
BITS	

8.5.1.51 Value of OTP Bank5 Word3 (SW GP2) (HW_OCOTP_SW_GP22)

8.5.1.51.1 Offset

Register	Offset
HW_OCOTP_SW_GP22	6B0h

8.5.1.51.2 Function

Shadowed memory mapped access to OTP Bank 5, word 3 (ADDR = 0x2b).

8.5.1.51.3 Diagram

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

8.5.1.51.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 3 (ADDR = 0x2b).
BITS	

8.5.1.52 Value of OTP Bank5 Word4 (SW GP2) (HW_OCOTP_SW_GP23)

8.5.1.52.1 Offset

Register	Offset
HW_OCOTP_SW_GP23	6C0h

8.5.1.52.2 Function

Shadowed memory mapped access to OTP Bank 5, word 4 (ADDR = 0x2c).

8.5.1.52.3 Diagram

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

8.5.1.52.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 4 (ADDR = 0x2c).
BITS	

8.5.1.53 Value of OTP Bank5 Word5 (Misc Conf) (HW_OCOTP_MISC_CONF0)

8.5.1.53.1 Offset

Register	Offset
HW_OCOTP_MISC_CO_NF0	6D0h

8.5.1.53.2 Function

Shadowed memory mapped access to OTP Bank 5, word 5 (ADDR = 0x2d).

8.5.1.53.3 Diagram

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

8.5.1.53.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 5 (ADDR = 0x2d).
BITS	

8.5.1.54 Value of OTP Bank5 Word6 (Misc Conf) (HW_OCOTP_MISC_CONF1)

8.5.1.54.1 Offset

Register	Offset
HW_OCOTP_MISC_CO_NF1	6E0h

8.5.1.54.2 Function

Shadowed memory mapped access to OTP Bank 5, word 6 (ADDR = 0x2e).

8.5.1.54.3 Diagram

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

8.5.1.54.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 6 (ADDR = 0x2e).
BITS	

8.5.1.55 Value of OTP Bank5 Word7 (SRK Revoke) (HW_OCOTP_SRK_REVOKED)

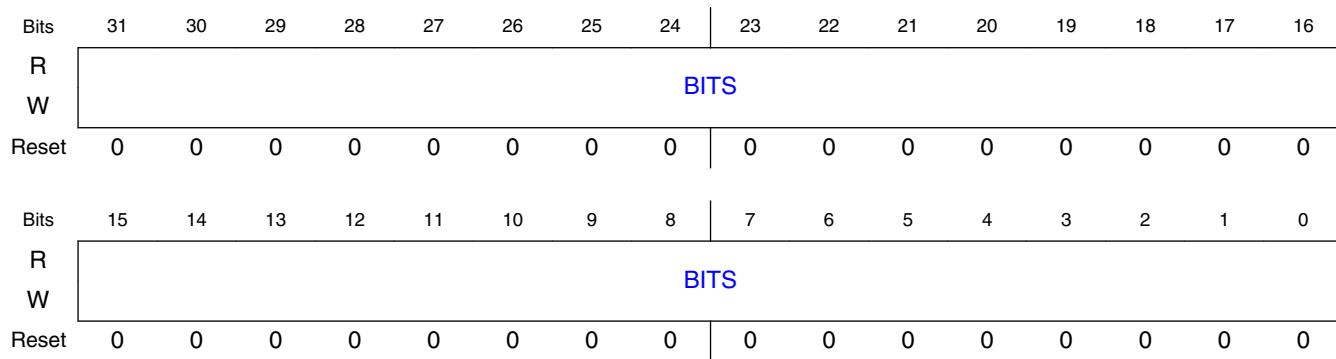
8.5.1.55.1 Offset

Register	Offset
HW_OCOTP_SRK_REVOKED	6F0h

8.5.1.55.2 Function

Shadowed memory mapped access to OTP Bank 5, word 7 (ADDR = 0x2f).

8.5.1.55.3 Diagram



8.5.1.55.4 Fields

Field	Function
31-0	Reflects value of OTP Bank 5, word 7 (ADDR = 0x2f).
BITS	

Chapter 9

True Random Number Generator (TRNG)

9.1 Chip-specific TRNG information

NOTE

TRNG Access Mode (MCTL[TRNG_ACC] bit, as well as related content) is not applicable on this device.

9.2 Standalone True Random Number Generator (SA-TRNG).

The Standalone True Random Number Generator (SA-TRNG) is a hardware accelerator module that generates a 512-bit entropy as needed by an entropy-consuming module or by other post-processing functions. A typical entropy consumer is a pseudo random-number generator (PRNG) that can be implemented to achieve both true randomness and cryptographic-strength random numbers using the TRNG output as its entropy seed. The PRNG is not part of this module.

The entropy generated by a TRNG is intended for direct use by functions that generate secret keys, per-message secrets, random challenges, and other similar quantities used in cryptographic algorithms. In each of these cases, it is important that a random number be difficult to guess or predict. It is important that a random number is at least as difficult to predict as it is difficult to break the cryptographic algorithm with which it is being used. This stringent requirement is particularly difficult to fulfill if the entropy source from a TRNG contains bias and/or correlation. To increase the trustworthiness/quality of the generated random data, PRNGs are often used to post process the output of a TRNG.

This document describes only the TRNG design functionality and usage.

Note that before entropy can be obtained from the TRNG, it must be initialized and instantiated in a particular mode by setting the appropriate TRNG registers.

Standalone True Random Number Generator (SA-TRNG).

The TRNG contains the following sub modules: IP Slave bus (SkyBlue bus) interface, the TRNG Core and the free running oscillator (OSC).

9.2.1 Standalone True Random Number Generator Block Diagram

The following figure is a top-level diagram of the True Random Number Generator.

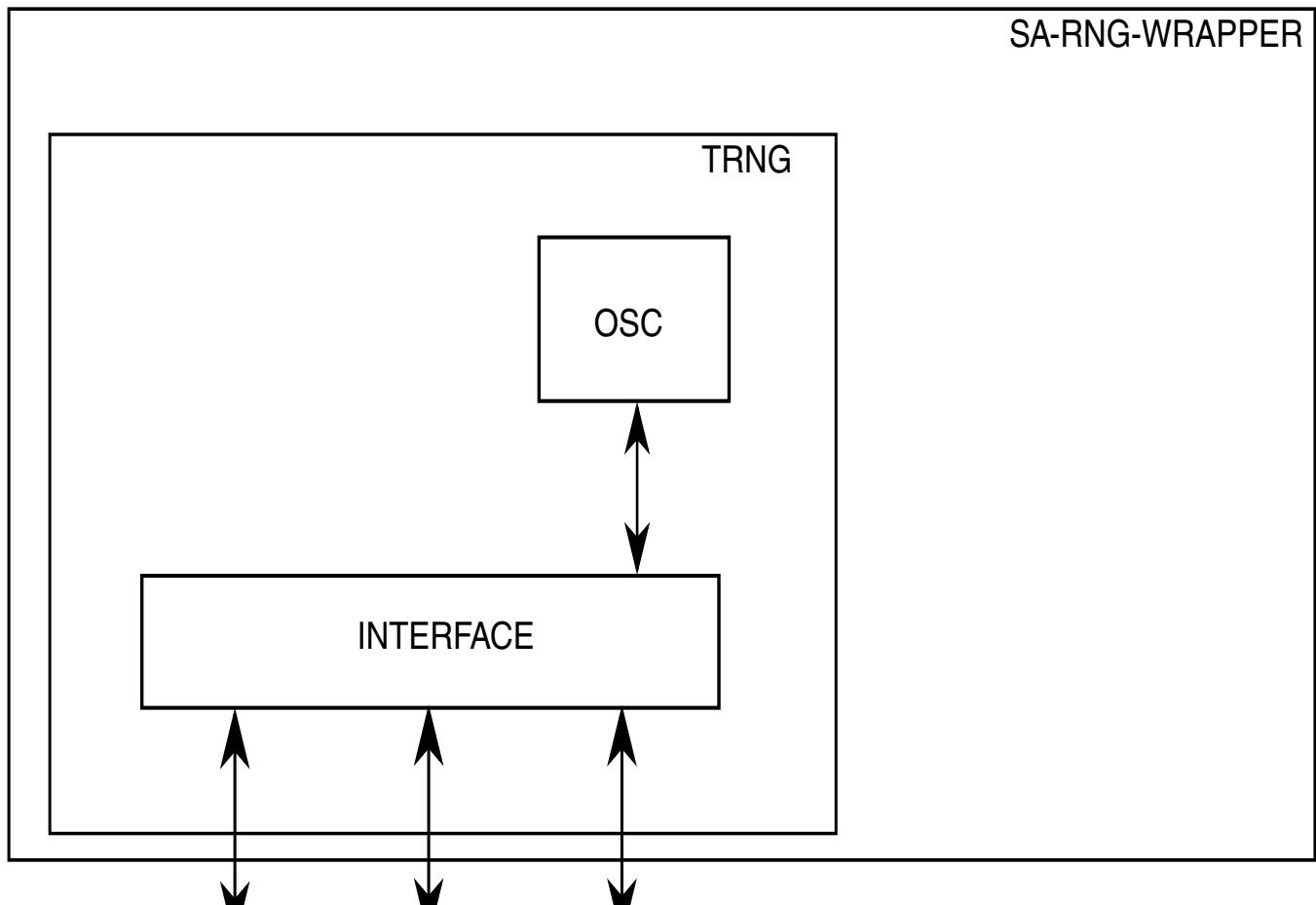


Figure 9-1. SA TRNG Block Diagram

9.2.2 TRNG Functional Description.

The TRNG consists of several functional sub-modules. Its overall functionality can be easily described from the top level in terms of generating entropy for seed generation. The functionality of each sub-module is briefly described in the following subsections.

TRNG is based on collecting bits from a random noise source. This random noise source is a ring oscillator that is sensitive to random noise (temperature variations, voltage variations, cross-talk and other random noise) within the device in which the TRNG is used. This noise causes various small changes in the period of the oscillator. Therefore, if the count of the ring oscillator clock cycles is sampled after a known period of time, this count will vary each time the sample is taken. By using the variance in this count over a large number of samples, random bits can be derived.

The TRNG comprises two entropy sources, each of which provides a single bit of output. Concatenated together, these 2 output bits are expected to provide 1 bit of entropy every 100 clock cycles. In addition to generating entropy, the TRNG also performs several statistical tests on its output.

9.2.3 SA-TRNG hardware functional description.

SA-TRNG functionality consists of several major subcomponents. This table describes these subcomponents.

Table 9-1. SA-TRNG subcomponents.

Description	Cross-reference(s)
Interfaces	
Register interface <ul style="list-style-type: none"> Used for access to configuration, control, status and debugging registers 	Register interface (IP Slave bus)
True Random Number Generator (TRNG)	Standalone True Random Number Generator (SA-TRNG).

9.2.3.1 Software Use Cases for the Stand Alone TRNG.

There are four things that a user (programmer/integrator) will want to do with a TRNG.

- Initialization.

Set up the parameters to proper values, and start generation of the first block of entropy. This is done once.

- Read entropy from the TRNG, and start generation of the next block of entropy.

This is done many times and is the normal flow of operation.

- Run a self-test on the TRNG, to assure proper continued operation.

This involves taking TRNG off-line, setting some self-test parameters, running TRNG, and then reading the statistical test registers, to see that they are within proper operation values. This may not be needed, as TRNG has built-in self-test.

- Off-line determination and checking of TRNG parameter values.

This is done in development in order to determine the proper initialization and self-test parameters. The TRNG is taken off-line. Test parameter values are written and entropy generation is started. If the statistical tests indicate poor operation (i.e., failing statistical tests), the entropy_delay value should be increased and entropy generation should be re-started. Every case is a variation of setting TRNG parameter values, starting or re-starting entropy generation and reading out the entropy. This process requires pausing or stopping and re-starting the TRNG.

The TRNG is designed to operate as a slave module on the standard IP Slave Bus. By understanding the TRNG register descriptions in "TRNG Register Descriptions" section below, the TRNG module can be controlled via the IP slave bus. In order to write to most TRNG registers, the MCTL register must be initialized in programming mode as described in the "TRNG Register Descriptions" section. At Power On Reset (POR) or asynchronous system hard reset, the TRNG resets to programming mode. TRNG will not generate entropy until it is out of programming mode (in run mode) and access to Entropy Registers have been enabled.

Here is an example program flow of using the TRNG.

- After POR the TRNG will be reset into programming mode with the OK to stop bit set(MCTL[TSTOP_OK]=1). The TRNG must be put into Run Mode for Entropy Generation to begin (MCTL[PRGM]=0). Using the default self test limits that exist after bootup, the entropy valid bit can be polled until asserted (MCTL[ENT_VAL]=1). Alternatively, if using the interrupt, and the interrupts are enabled via the INT_MASK register and the ipi_rng_int_b is asserted when MCTL[ENT_VAL]=1.
- After the polling completes, the 512-bit entropy generated by the TRNG can be read. The values can be read in any order from entropy register 0 to register 15 (ENT0 to ENT 15). After reading ENT 15, the old entropy value is reset and a new entropy value is generated.

NOTE

Reading ENT 15 always resets the entropy, so should always be read last.

- You can poll again for the new entropy value or you can use the Interrupt Status Register to handle reading the entropy values when the entropy valid interrupt is triggered.

- The interrupt can be masked or cleared as needed. See the Interrupt Status Register description.
- To change the self-test limits, the seed counters, how fast the entropy is generated, and how entropy is sampled, see the register description section. In particular, see the the TRNG Frequency Count Minimum Limit Register (FRQMIN), the seed control register (SCML), the statistical run length registers, and other parameter registers.
- Once in Run Mode, the entropy is re-generated automatically after ENT 15 is read. Setting the TRNG back to programming mode (MCTL[PRGM]=1) also achieves the purpose of stopping entropy generation.

9.2.4 Another TRNG usage example.

The TRNG can be used by a post processing pseudo-random number generator function. For example, TRNG can be used to seed a hardware or software based implementation of a DRBG defined by SP800-90.

9.2.5 TRNG register descriptions

All accesses of undefined addresses always return zero and assert IPS transfer error. Writes to undefined and read-only addresses are ignored. Undefined addresses are those undocumented, protected or reserved addresses within and outside the range of the addresses defined in the memory map below. The register addresses shown in the Memory Map below represents how these registers are accessed over the register bus as 32-bit words. Accesses to registers must use full-word (32-bit) reads or writes.

The format and fields in each TRNG register are defined below. Some of the register format figures apply to several different registers. In such cases a different register name will be associated with each of the register offset addresses that appear at the top of the register format figure. Although these registers share the same format, they are independent registers. In addition, many registers can be accessed at multiple addresses. In these cases there will be a single register name and the list of addresses at which that register is accessible will be indicated as aliases. Unless noted in the individual register descriptions, registers are reset only at Power-On Reset (POR) or with an asynchronous system hard reset.

9.2.5.1 TRNG Memory map

TRNG base address: 400C_C000h

Standalone True Random Number Generator (SA-TRNG).

Offset	Register	Width (In bits)	Access	Reset value
0h	Miscellaneous Control Register (MCTL)	32	RW	0001_2001h
4h	Statistical Check Miscellaneous Register (SCMISC)	32	RW	0001_0022h
8h	Poker Range Register (PKRRNG)	32	RW	0000_09A3h
Ch	Poker Maximum Limit Register (PKRMAX)	32	RW	0000_6920h
Ch	Poker Square Calculation Result Register (PKRSQ)	32	RO	0000_0000h
10h	Seed Control Register (SDCTL)	32	RW	0C80_09C4h
14h	Sparse Bit Limit Register (SBLIM)	32	RW	0000_003Fh
14h	Total Samples Register (TOTSAM)	32	RO	0000_0000h
18h	Frequency Count Minimum Limit Register (FRQMIN)	32	RW	0000_0640h
1Ch	Frequency Count Register (FRQCNT)	32	RO	0000_0000h
1Ch	Frequency Count Maximum Limit Register (FRQMAX)	32	RW	0000_6400h
20h	Statistical Check Monobit Count Register (SCMC)	32	RO	0000_0000h
20h	Statistical Check Monobit Limit Register (SCML)	32	RW	010C_0568h
24h	Statistical Check Run Length 1 Count Register (SCR1C)	32	RO	0000_0000h
24h	Statistical Check Run Length 1 Limit Register (SCR1L)	32	RW	00B2_0195h
28h	Statistical Check Run Length 2 Count Register (SCR2C)	32	RO	0000_0000h
28h	Statistical Check Run Length 2 Limit Register (SCR2L)	32	RW	007A_00DCh
2Ch	Statistical Check Run Length 3 Count Register (SCR3C)	32	RO	0000_0000h
2Ch	Statistical Check Run Length 3 Limit Register (SCR3L)	32	RW	0058_007Dh
30h	Statistical Check Run Length 4 Count Register (SCR4C)	32	RO	0000_0000h
30h	Statistical Check Run Length 4 Limit Register (SCR4L)	32	RW	0040_004Bh
34h	Statistical Check Run Length 5 Count Register (SCR5C)	32	RO	0000_0000h
34h	Statistical Check Run Length 5 Limit Register (SCR5L)	32	RW	002E_002Fh
38h	Statistical Check Run Length 6+ Count Register (SCR6PC)	32	RO	0000_0000h
38h	Statistical Check Run Length 6+ Limit Register (SCR6PL)	32	RW	002E_002Fh
3Ch	Status Register (STATUS)	32	RO	0000_0000h
40h - 7Ch	Entropy Read Register (ENT0 - ENT15)	32	RO	0000_0000h
80h	Statistical Check Poker Count 1 and 0 Register (PKRCNT10)	32	RO	0000_0000h
84h	Statistical Check Poker Count 3 and 2 Register (PKRCNT32)	32	RO	0000_0000h
88h	Statistical Check Poker Count 5 and 4 Register (PKRCNT54)	32	RO	0000_0000h
8Ch	Statistical Check Poker Count 7 and 6 Register (PKRCNT76)	32	RO	0000_0000h
90h	Statistical Check Poker Count 9 and 8 Register (PKRCNT98)	32	RO	0000_0000h
94h	Statistical Check Poker Count B and A Register (PKRCNTBA)	32	RO	0000_0000h
98h	Statistical Check Poker Count D and C Register (PKRCNTDC)	32	RO	0000_0000h
9Ch	Statistical Check Poker Count F and E Register (PKRCNTFE)	32	RO	0000_0000h
A0h	Security Configuration Register (SEC_CFG)	32	RW	0000_0000h
A4h	Interrupt Control Register (INT_CTRL)	32	RW	0000_0007h
A8h	Mask Register (INT_MASK)	32	RW	0000_0000h
ACh	Interrupt Status Register (INT_STATUS)	32	RO	0000_0000h
F0h	Version ID Register (MS) (VID1)	32	RO	0030_0301h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
F4h	Version ID Register (LS) (VID2)	32	RO	0000_0000h

9.2.5.2 Miscellaneous Control Register (MCTL)

9.2.5.2.1 Offset

Register	Offset
MCTL	0h

9.2.5.2.2 Function

This register is intended to be used for programming, configuring and testing the RNG. It is the main register to read/write, in order to enable Entropy generation, to stop entropy generation and to block access to entropy registers. This is done via the special TRNG_ACC and PRGM bit below.

The Miscellaneous Control Register is a read/write register used to control the RNG's True Random Number Generator (TRNG) access, operation and test.

NOTE

Note that in many cases two RNG registers share the same address, and a particular register at the shared address is selected based upon the value in the PRGM field of the MCTL register.

9.2.5.2.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	LRUN_CONT	TSTOP_OK	ER_R	TST_OUT	ENT_VAL	FCT_VAL	FCT_FAIL	FOR_SCLK	RST_DEF	TRNG_ACC	UNUSED4	CSC_DIV	SAMP_MODE		
W				W1C						0	0	0	0	0	0	
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

9.2.5.2.4 Fields

Field	Function
31-17 —	Reserved.
16 PRGM	Programming Mode Select. When this bit is 1, the TRNG is in Program Mode, otherwise it is in Run Mode. No Entropy value will be generated while the TRNG is in Program Mode. Note that different RNG registers are accessible at the same address depending on whether PRGM is set to 1 or 0. This is noted in the RNG register descriptions.
15 —	Reserved.
14 LRUN_CONT	Long run count continues between entropy generations. If set to 1, the internal test's long_run count at the start of an entropy generation cycle will continue where it left off at the end of the previous entropy generation. If set to 0, then normal LRUN count is restarted for each entropy re-generation.
13 TSTOP_OK	TRNG_OK_TO_STOP. Software can check that this bit is a 1 before transitioning TRNG to low power mode (TRNG clock stopped). TRNG turns on the TRNG free-running ring oscillator whenever new entropy is being generated and turns off the ring oscillator when entropy generation is complete. If the TRNG clock is stopped while the TRNG ring oscillator is running, the oscillator will continue running even though the TRNG clock is stopped. To make sure the ring oscillator is stopped, assert ipg_stop input and verify ipg_rng_stop_ack is asserted after at least two (2) clock cycles. TSTOP_OK is asserted when the TRNG ring oscillator is not running. This helps for cases where you want to stop the TRNG TRNG clock without having access to ipg_stop nor ipg_rng_stop_ack.
12 ERR	Read: Error status. 1 = error detected. 0 = no error. Write: Write 1 to clear errors. Writing 0 has no effect.
11 TST_OUT	Read only: Test point inside ring oscillator.

Table continues on the next page...

Field	Function
10 ENT_VAL	Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared at most one (1) bus clock cycle after reading the ENT15 register. (ENT0 through ENT14 should be read before reading ENT15).
9 FCT_VAL	Read only: Frequency Count Valid. Indicates that a valid frequency count may be read from FRQCN.
8 FCT_FAIL	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator. This error may be cleared by writing a 1 to the ERR bit.
7 FOR_SCLK	Force System Clock. If set, the system clock is used to operate the TRNG, instead of the ring oscillator. This is for test use only, and indeterminate results may occur. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. This bit is cleared by writing the RST_DEF bit to 1.
6 RST_DEF	Reset Defaults. Writing a 1 to this bit clears various TRNG registers, and bits within registers, to their default state. This bit is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this bit. Reading this bit always produces a 0.
5 TRNG_ACC	TRNG Access Mode. If this bit is set to 1, the TRNG will generate an Entropy value that can be read via the ENT0-ENT15 registers. The Entropy value may be read once the ENT VAL bit is asserted. Also see ENTa register descriptions (For a = 0 to 15).
4 UNUSED4	This bit is unused. Always reads zero.
3-2 OSC_DIV	Oscillator Divide. Determines the amount of dividing done to the ring oscillator before it is used by the TRNG. This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously to writing this field. This field is cleared to the default POR value by writing the RST_DEF bit to 1. 00b - use ring oscillator with no divide 01b - use ring oscillator divided-by-2 10b - use ring oscillator divided-by-4 11b - use ring oscillator divided-by-8
1-0 SAMP_MODE	Sample Mode. Determines the method of sampling the ring oscillator while generating the Entropy value: This field is writable only if PRGM bit is 1, or PRGM bit is being written to 1 simultaneously with writing this field. This field is cleared to the POR default value by writing the RST_DEF bit to 1. 00b - use Von Neumann data into both Entropy shifter and Statistical Checker 01b - use raw data into both Entropy shifter and Statistical Checker 10b - use Von Neumann data into Entropy shifter. Use raw data into Statistical Checker 11b - undefined/reserved.

9.2.5.3 Statistical Check Miscellaneous Register (SCMISC)

9.2.5.3.1 Offset

Register	Offset
SCMISC	4h

9.2.5.3.2 Function

The Statistical Check Miscellaneous Register contains the Long Run Maximum Limit value and the Retry Count value. This register is accessible only when the MCTL[PRGM] bit is 1, otherwise this register will read zeroes, and cannot be written.

9.2.5.3.3 Diagram

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

9.2.5.3.4 Fields

Field	Function
31-20 —	Reserved.
19-16 RTY_CT	RETRY COUNT. If a statistical check fails during the TRNG Entropy Generation, the RTY_CT value indicates the number of times a retry should occur before generating an error. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-8 —	Reserved.
7-0 LRUN_MAX	LONG RUN MAX LIMIT. This value is the largest allowable number of consecutive samples of all 1, or all 0, that is allowed during the Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the POR reset value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.4 Poker Range Register (PKRRNG)

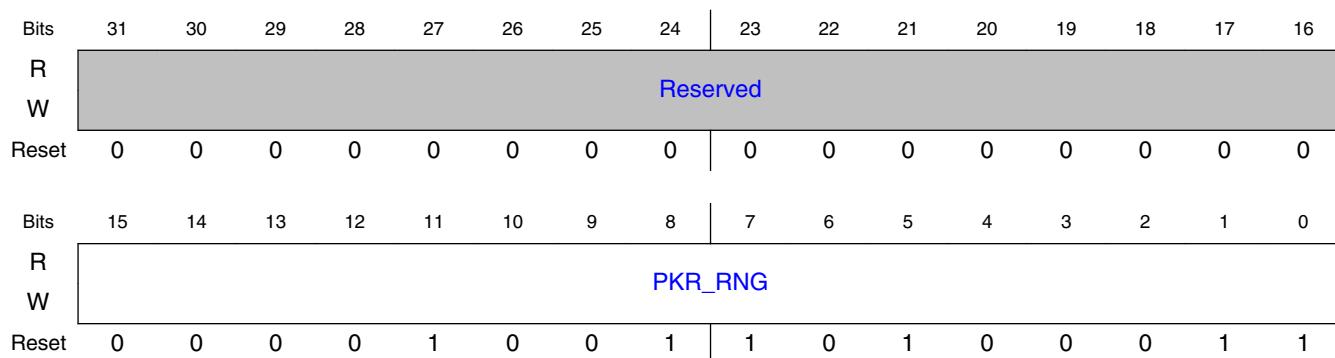
9.2.5.4.1 Offset

Register	Offset
PKRRNG	8h

9.2.5.4.2 Function

The Poker Range Register defines the difference between the TRNG Poker Maximum Limit and the minimum limit. These limits are used during the TRNG Statistical Check Poker Test.

9.2.5.4.3 Diagram



9.2.5.4.4 Fields

Field	Function
31-16	Reserved. Always 0.
—	
15-0 PKR_RNG	Poker Range. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum is programmed in the PKRMAX[PKR_MAX] register, and the minimum is derived by subtracting the PKR_RNG value from the programmed maximum value. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the minimum allowable Poker result is PKR_MAX - PKR_RNG + 1.

9.2.5.5 Poker Square Calculation Result Register (PKRSQ)

9.2.5.5.1 Offset

Register	Offset	Description
PKRSQ	Ch	Accessible at this address when MCTL[PRGM] = 0

9.2.5.5.2 Function

The Poker Square Calculation Result Register is a read-only register used to read the result of the TRNG Statistical Check Poker Test's Square Calculation. This test starts with the PKRMAX value and decreases towards a final result, which is read here. For the Poker Test to pass, this final result must be less than the programmed PKRRNG value. Note that this offset (0x0C) is used as PKRMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as PKRSQ readback register, as described here.

9.2.5.5.3 Diagram

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

9.2.5.5.4 Fields

Field	Function
31-24 —	Reserved.
23-0 PKR_SQ	Poker Square Calculation Result. During the TRNG Statistical Checks, a "Poker Test" is run which starts with the value PKRMAX[PKR_MAX]. This value decreases according to a "sum of squares" algorithm, and must remain greater than zero, but less than the PKRRNG[PKR_RNG] limit. The resulting value may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Poker Test Maximum Limit in register PKRMAX, as defined in the previous section.

9.2.5.6 Poker Maximum Limit Register (PKRMAX)

9.2.5.6.1 Offset

Register	Offset	Description
PKRMAX	Ch	Accessible at this address when MCTL[PRGM] = 1

9.2.5.6.2 Function

The Poker Maximum Limit Register defines Maximum Limit allowable during the TRNG Statistical Check Poker Test. Note that this offset (0x0C) is used as PKRMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as the PKRSQ readback register.

9.2.5.6.3 Diagram

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

9.2.5.6.4 Fields

Field	Function
31-24 —	Reserved.
23-0 PKR_MAX	Poker Maximum Limit. During the TRNG Statistical Checks, a "Poker Test" is run which requires a maximum and minimum limit. The maximum allowable result is programmed in the PKRMAX[PKR_MAX] register. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that the PKRMAX and PKRRNG registers combined are used to define the minimum allowable Poker result, which is PKR_MAX - PKR_RNG + 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Poker Test Square Calculation result in register PKRSQ, as defined in the following section.

9.2.5.7 Seed Control Register (SDCTL)

9.2.5.7.1 Offset

Register	Offset
SDCTL	10h

9.2.5.7.2 Function

The Seed Control Register contains two fields. One field defines the length (in system clocks) of each Entropy sample (ENT_DLY), and the other field indicates the number of samples that will be taken during each TRNG Entropy generation (SAMP_SIZE).

9.2.5.7.3 Diagram

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

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

9.2.5.7.4 Fields

Field	Function
31-16 ENT_DLY	Entropy Delay. Defines the length (in system clocks) of each Entropy sample taken. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to its reset value at POR.
15-0 SAMP_SIZE	Sample Size. Defines the total number of Entropy samples that will be taken during Entropy generation. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.8 Sparse Bit Limit Register (SBLIM)

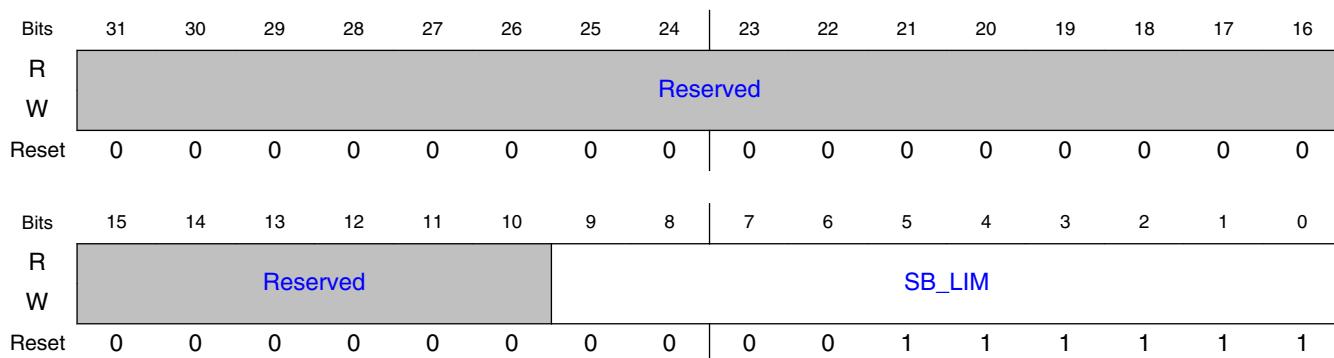
9.2.5.8.1 Offset

Register	Offset	Description
SBLIM	14h	Accessible at this address when MCTL[PRGM] = 1

9.2.5.8.2 Function

The Sparse Bit Limit Register is used when Von Neumann sampling is selected during Entropy Generation. It defines the maximum number of consecutive Von Neumann samples which may be discarded before an error is generated. Note that this address (0x14) is used as SBLIM only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as TOTSAM readback register.

9.2.5.8.3 Diagram



9.2.5.8.4 Fields

Field	Function
31-10 —	Reserved. Always 0.
9-0 SB_LIM	Sparse Bit Limit. During Von Neumann sampling (if enabled by MCTL[SAMP_MODE]), samples are discarded if two consecutive raw samples are both 0 or both 1. If this discarding occurs for a long period of time, it indicates that there is insufficient Entropy. The Sparse Bit Limit defines the maximum number of consecutive samples that may be discarded before an error is generated. This field is writable only if MCTL[PRGM] bit is 1. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Total Samples count in register TOTSAM, as defined in the following section.

9.2.5.9 Total Samples Register (TOTSAM)

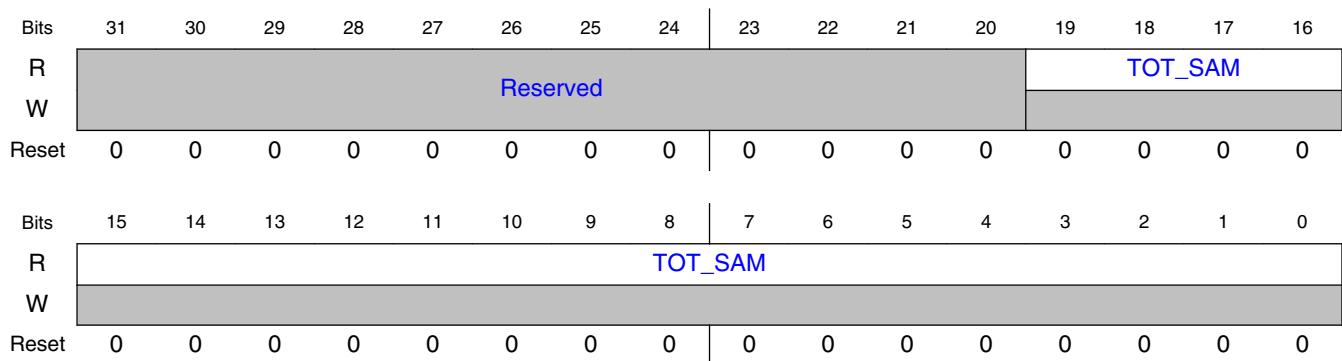
9.2.5.9.1 Offset

Register	Offset	Description
TOTSAM	14h	Accessible at this address when MCTL[PRGM] = 0

9.2.5.9.2 Function

The Total Samples Register is a read-only register used to read the total number of samples taken during Entropy generation. It is used to give an indication of how often a sample is actually used during Von Neumann sampling. Note that this offset (0x14) is used as SBLIM if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as TOTSAM readback register, as described here.

9.2.5.9.3 Diagram



9.2.5.9.4 Fields

Field	Function
31-20 —	Reserved. Always 0.
19-0 TOT_SAM	Total Samples. During Entropy generation, the total number of raw samples is counted. This count is useful in determining how often a sample is used during Von Neumann sampling. The count may be read through this register, if MCTL[PRGM] bit is 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Sparse Bit Limit in register SBLIM, as defined in the previous section.

9.2.5.10 Frequency Count Minimum Limit Register (FRQMIN)

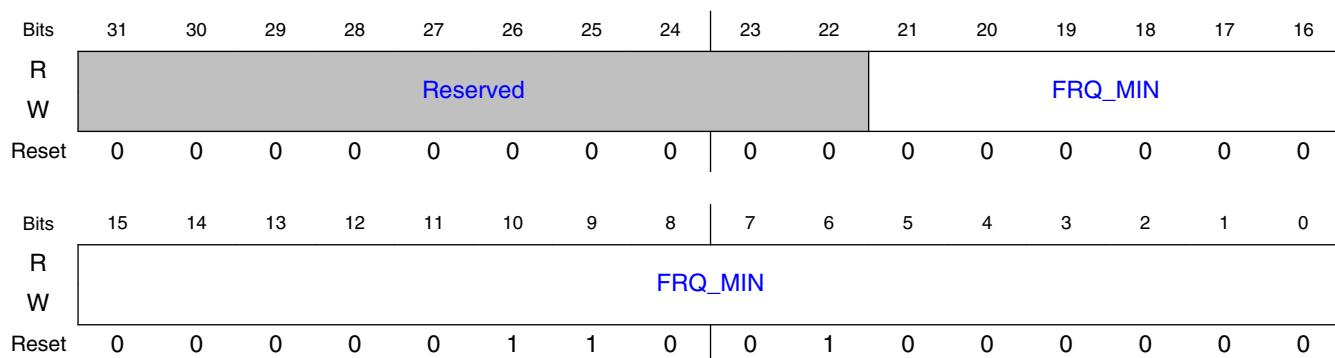
9.2.5.10.1 Offset

Register	Offset
FRQMIN	18h

9.2.5.10.2 Function

The Frequency Count Minimum Limit Register defines the minimum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is less than this programmed minimum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated.

9.2.5.10.3 Diagram



9.2.5.10.4 Fields

Field	Function
31-22	Reserved. Always 0.
—	
21-0 FRQ_MIN	Frequency Count Minimum Limit. Defines the minimum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field will read zeroes if MCTL[PRGM] = 0. This field is cleared to its reset value at POR.

9.2.5.11 Frequency Count Maximum Limit Register (FRQMAX)

9.2.5.11.1 Offset

Register	Offset	Description
FRQMAX	1Ch	Accessible at this address when MCTL[PRGM] = 1

9.2.5.11.2 Function

The Frequency Count Maximum Limit Register defines the maximum allowable count taken by the Entropy sample counter during each Entropy sample. During any sample period, if the count is greater than this programmed maximum, a Frequency Count Fail is flagged in MCTL[FCT_FAIL] and an error is generated. Note that this address (001C) is used as FRQMAX only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as FRQCNT readback register.

9.2.5.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								FRQ_MAX							
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<hr/>																
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRQ_MAX															
W	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
Reset	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0

9.2.5.11.4 Fields

Field	Function
31-22	Reserved. Always 0.
—	
21-0 FRQ_MAX	Frequency Counter Maximum Limit. Defines the maximum allowable count taken during each entropy sample. This field is writable only if MCTL[PRGM] bit is 1. This field is cleared to its reset value at POR. Note that if MCTL[PRGM] bit is 0, this register address is used to read the Frequency Count result in register FRQCNT, as defined in the FRQ_CT field.

9.2.5.12 Frequency Count Register (FRQCNT)

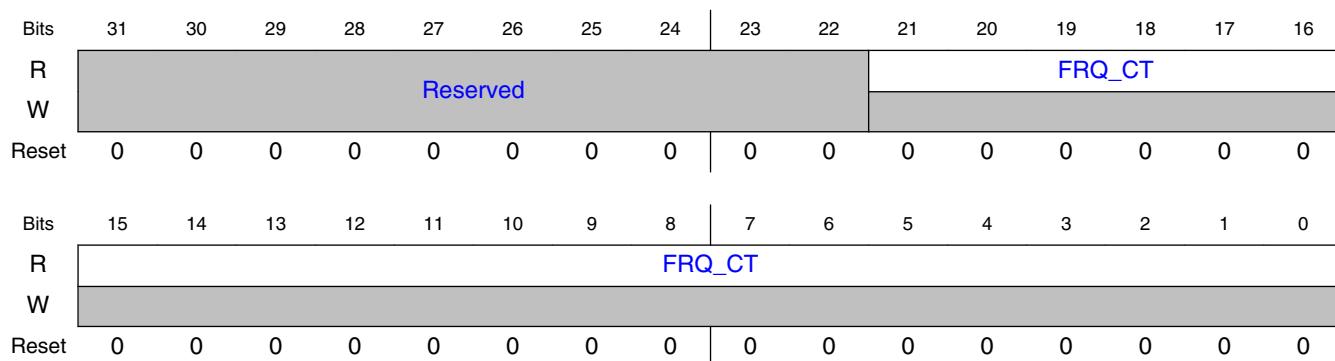
9.2.5.12.1 Offset

Register	Offset	Description
FRQCNT	1Ch	Accessible at this address when MCTL[PRGM] = 0

9.2.5.12.2 Function

The Frequency Count Register is a read-only register used to read the frequency counter within the TRNG entropy generator. It will read all zeroes unless PRGM[MCTL] = 0 and MCTL[TRNG_ACC] = 1. Note that this offset (0x1C) is used as FRQMAX if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as live FRQCNT readback register, as described here. This register must be read once, as a full 32-bit. A subsequent read is only valid when MCTL[FCT_VAL] bit is HIGH. A read clears this register. The next read when MCTL[FCT_VAL] bit is HIGH must be treated as a different value from the previous one, even though the counts might be the same.

9.2.5.12.3 Diagram



9.2.5.12.4 Fields

Field	Function
31-22	Reserved. Always 0.
—	
21-0 FRQ_CT	Frequency Count. If MCTL[TRNG_ACC] = 1, Reads a sample frequency count taken during entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Frequency Maximum Limit in register FRQMAX, as defined in the FRQ_MAX field.

9.2.5.13 Statistical Check Monobit Count Register (SCMC)

9.2.5.13.1 Offset

Register	Offset	Description
SCMC	20h	Accessible at this address when MCTL[PRGM] = 0

9.2.5.13.2 Function

The Statistical Check Monobit Count Register is a read-only register used to read the final monobit count after entropy generation. This counter starts with the value in SCML[MONO_MAX], and is decremented each time a one is sampled. Note that this offset (0x20) is used as SCML if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register, as described here.

9.2.5.13.3 Diagram



9.2.5.13.4 Fields

Field	Function
31-16 —	Reserved. Always 0.
15-0 MONO_CT	Monobit Count. Reads the final Monobit count after entropy generation. Requires MCTL[PRGM] = 0. Note that if MCTL[PRGM] bit is 1, this register address is used to access the Statistical Check Monobit Limit in register SCML, as defined in the previous section.

9.2.5.14 Statistical Check Monobit Limit Register (SCML)

9.2.5.14.1 Offset

Register	Offset	Description
SCML	20h	Accessible at this address when MCTL[PRGM] = 1

9.2.5.14.2 Function

The Statistical Check Monobit Limit Register defines the allowable maximum and minimum number of ones/zero detected during entropy generation. To pass the test, the number of ones/zeros generated must be less than the programmed maximum value, and the number of ones/zeros generated must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x20) is used as SCML only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCMC readback register.

9.2.5.14.3 Diagram

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

9.2.5.14.4 Fields

Field	Function
31-16 MONO_RNG	Monobit Range. The number of ones/zeros detected during entropy generation must be greater than MONO_MAX - MONO_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-0 MONO_MAX	Monobit Maximum Limit. Defines the maximum allowable count taken during entropy generation. The number of ones/zeros detected during entropy generation must be less than MONO_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.15 Statistical Check Run Length 1 Limit Register (SCR1L)

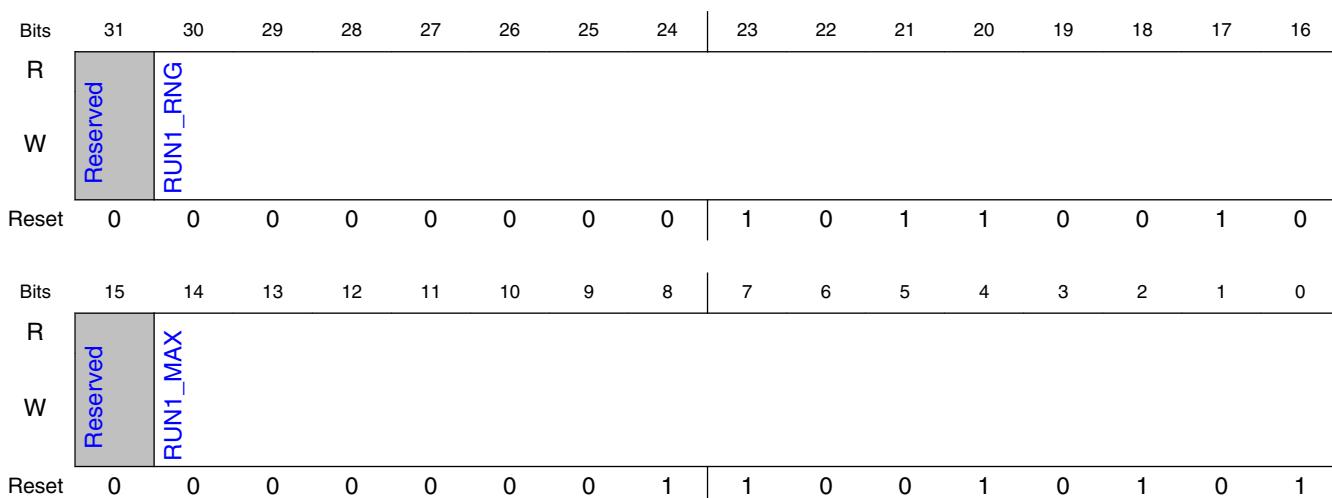
9.2.5.15.1 Offset

Register	Offset	Description
SCR1L	24h	Accessible at this address when MCTL[PRGM] = 1

9.2.5.15.2 Function

The Statistical Check Run Length 1 Limit Register defines the allowable maximum and minimum number of runs of length 1 detected during entropy generation. To pass the test, the number of runs of length 1 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 1 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x24) is used as SCR1L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR1C readback register.

9.2.5.15.3 Diagram



9.2.5.15.4 Fields

Field	Function
31	Reserved. Always 0.
—	
30-16 RUN1_RNG	Run Length 1 Range. The number of runs of length 1 (for both 0 and 1) detected during entropy generation must be greater than RUN1_MAX - RUN1_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15	Reserved. Always 0.
—	
14-0 RUN1_MAX	Run Length 1 Maximum Limit. Defines the maximum allowable runs of length 1 (for both 0 and 1) detected during entropy generation. The number of runs of length 1 detected during entropy generation must be less than RUN1_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.16 Statistical Check Run Length 1 Count Register (SCR1C)

9.2.5.16.1 Offset

Register	Offset	Description
SCR1C	24h	Accessible at this address when MCTL[PRGM] = 0]

9.2.5.16.2 Function

The Statistical Check Run Length 1 Counters Register is a read-only register used to read the final Run Length 1 counts after entropy generation. These counters start with the value in SCR1L[RUN1_MAX]. The R1_1_CT decrements each time a single one is sampled (preceded by a zero and followed by a zero). The R1_0_CT decrements each time a single zero is sampled (preceded by a one and followed by a one). Note that this offset (0x24) is used as SCR1L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR1C readback register, as described here.

9.2.5.16.3 Diagram

Bits	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	Reserved																
W	R1_1_CT																
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	Reserved																
W	R1_0_CT																
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

9.2.5.16.4 Fields

Field	Function
31 —	Reserved. Always 0.
30-16 R1_1_CT	Runs of One, Length 1 Count. Reads the final Runs of Ones, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.
15 —	Reserved. Always 0.
14-0 R1_0_CT	Runs of Zero, Length 1 Count. Reads the final Runs of Zeroes, length 1 count after entropy generation. Requires MCTL[PRGM] = 0.

9.2.5.17 Statistical Check Run Length 2 Count Register (SCR2C)

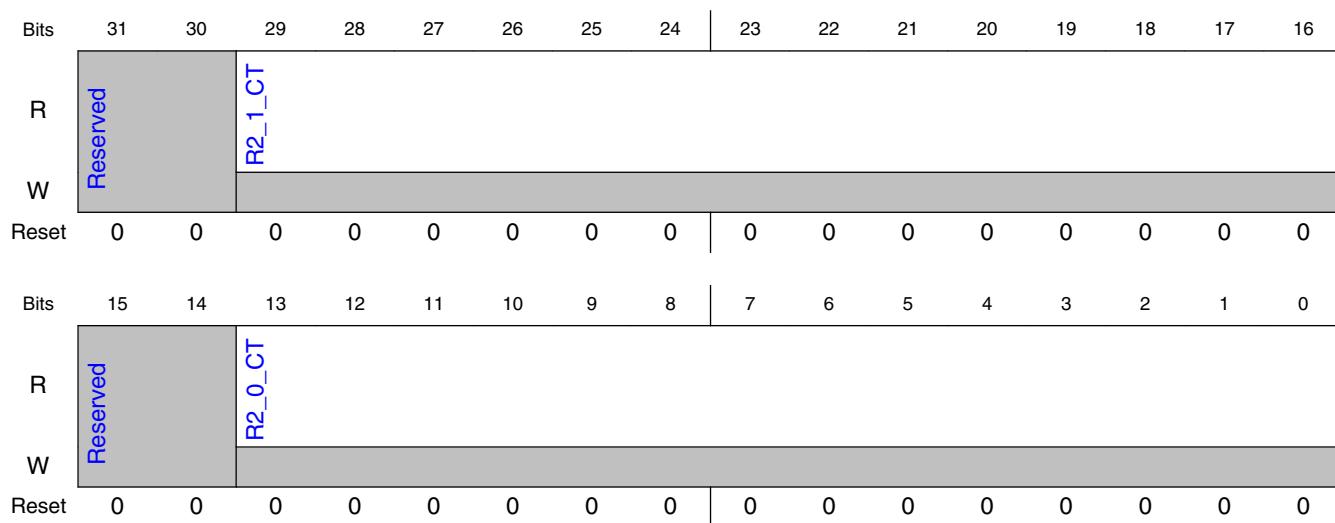
9.2.5.17.1 Offset

Register	Offset	Description
SCR2C	28h	Accessible at this address when MCTL[PRGM] = 0

9.2.5.17.2 Function

The Statistical Check Run Length 2 Counters Register is a read-only register used to read the final Run Length 2 counts after entropy generation. These counters start with the value in SCR2L[RUN2_MAX]. The R2_1_CT decrements each time two consecutive ones are sampled (preceded by a zero and followed by a zero). The R2_0_CT decrements each time two consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x28) is used as SCR2L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR2C readback register, as described here.

9.2.5.17.3 Diagram



9.2.5.17.4 Fields

Field	Function
31-30	Reserved. Always 0.
—	
29-16 R2_1_CT	Runs of One, Length 2 Count. Reads the final Runs of Ones, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.
15-14	Reserved. Always 0.
—	
13-0 R2_0_CT	Runs of Zero, Length 2 Count. Reads the final Runs of Zeroes, length 2 count after entropy generation. Requires MCTL[PRGM] = 0.

9.2.5.18 Statistical Check Run Length 2 Limit Register (SCR2L)

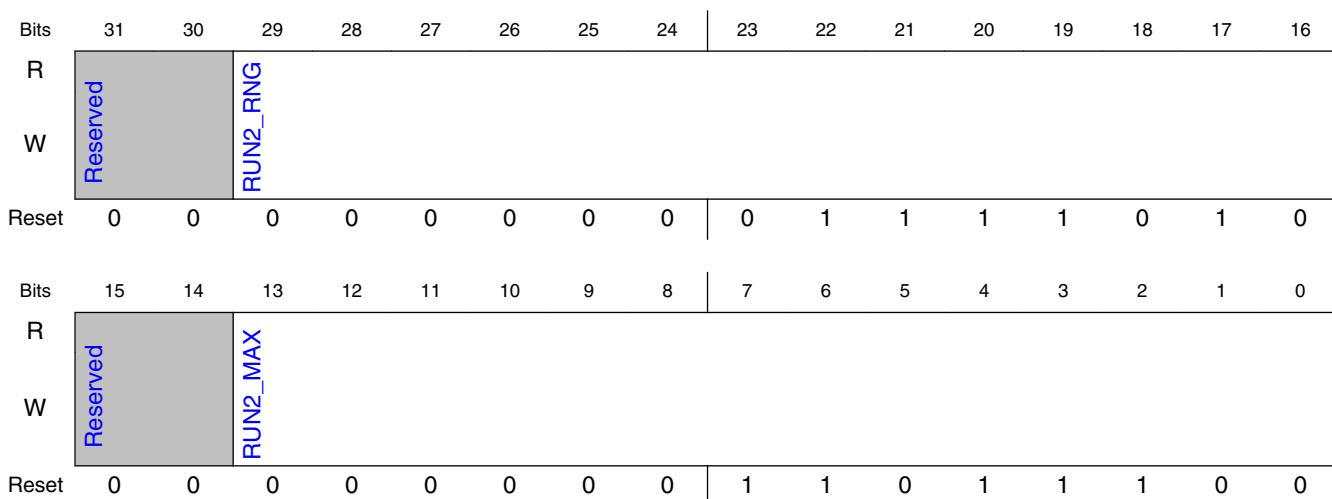
9.2.5.18.1 Offset

Register	Offset	Description
SCR2L	28h	Accessible at this address when MCTL[PRGM] = 1

9.2.5.18.2 Function

The Statistical Check Run Length 2 Limit Register defines the allowable maximum and minimum number of runs of length 2 detected during entropy generation. To pass the test, the number of runs of length 2 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 2 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x28) is used as SCR2L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR2C readback register.

9.2.5.18.3 Diagram



9.2.5.18.4 Fields

Field	Function
31-30	Reserved. Always 0.

Table continues on the next page...

Field	Function
—	
29-16 RUN2_RNG	Run Length 2 Range. The number of runs of length 2 (for both 0 and 1) detected during entropy generation must be greater than RUN2_MAX - RUN2_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-14 —	Reserved. Always 0.
13-0 RUN2_MAX	Run Length 2 Maximum Limit. Defines the maximum allowable runs of length 2 (for both 0 and 1) detected during entropy generation. The number of runs of length 2 detected during entropy generation must be less than RUN2_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.19 Statistical Check Run Length 3 Limit Register (SCR3L)

9.2.5.19.1 Offset

Register	Offset	Description
SCR3L	2Ch	Accessible at this address when MCTL[PRGM] = 1]

9.2.5.19.2 Function

The Statistical Check Run Length 3 Limit Register defines the allowable maximum and minimum number of runs of length 3 detected during entropy generation. To pass the test, the number of runs of length 3 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 3 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x2C) is used as SCR3L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR3C readback register.

9.2.5.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			RUN3_RNG												
W	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			RUN3_MAX												
W	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1

9.2.5.19.4 Fields

Field	Function
31-29 —	Reserved. Always 0.
28-16 RUN3_RNG	Run Length 3 Range. The number of runs of length 3 (for both 0 and 1) detected during entropy generation must be greater than RUN3_MAX - RUN3_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-13 —	Reserved. Always 0.
12-0 RUN3_MAX	Run Length 3 Maximum Limit. Defines the maximum allowable runs of length 3 (for both 0 and 1) detected during entropy generation. The number of runs of length 3 detected during entropy generation must be less than RUN3_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.20 Statistical Check Run Length 3 Count Register (SCR3C)

9.2.5.20.1 Offset

Register	Offset	Description
SCR3C	2Ch	Accessible at this address when MCTL[PRGM] = 0]

9.2.5.20.2 Function

The Statistical Check Run Length 3 Counters Register is a read-only register used to read the final Run Length 3 counts after entropy generation. These counters start with the value in SCR3L[RUN3_MAX]. The R3_1_CT decrements each time three consecutive

ones are sampled (preceded by a zero and followed by a zero). The R3_0_CT decrements each time three consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x2C) is used as SCR3L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR3C readback register, as described here.

9.2.5.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			R3_1_CT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			R3_0_CT												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.20.4 Fields

Field	Function
31-29	Reserved. Always 0.
—	
28-16 R3_1_CT	Runs of Ones, Length 3 Count. Reads the final Runs of Ones, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.
15-13	Reserved. Always 0.
—	
12-0 R3_0_CT	Runs of Zeroes, Length 3 Count. Reads the final Runs of Zeroes, length 3 count after entropy generation. Requires MCTL[PRGM] = 0.

9.2.5.21 Statistical Check Run Length 4 Count Register (SCR4C)

9.2.5.21.1 Offset

Register	Offset	Description
SCR4C	30h	Accessible at this address when MCTL[PRGM] = 0

9.2.5.21.2 Function

The Statistical Check Run Length 4 Counters Register is a read-only register used to read the final Run Length 4 counts after entropy generation. These counters start with the value in SCR4L[RUN4_MAX]. The R4_1_CT decrements each time four consecutive ones are sampled (preceded by a zero and followed by a zero). The R4_0_CT decrements each time four consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x30) is used as SCR4L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR4C readback register, as described here.

9.2.5.21.3 Diagram

Bits	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	Reserved				R4_1_CT												
W																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	Reserved				R4_0_CT												
W																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

9.2.5.21.4 Fields

Field	Function
31-28	Reserved. Always 0.
—	
27-16 R4_1_CT	Runs of One, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.
15-12	Reserved. Always 0.
—	
11-0 R4_0_CT	Runs of Zero, Length 4 Count. Reads the final Runs of Ones, length 4 count after entropy generation. Requires MCTL[PRGM] = 0.

9.2.5.22 Statistical Check Run Length 4 Limit Register (SCR4L)

9.2.5.22.1 Offset

Register	Offset	Description
SCR4L	30h	Accessible at this address when MCTL[PRGM] = 1

9.2.5.22.2 Function

The Statistical Check Run Length 4 Limit Register defines the allowable maximum and minimum number of runs of length 4 detected during entropy generation. To pass the test, the number of runs of length 4 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 4 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x30) is used as SCR4L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR4C readback register.

9.2.5.22.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				RUN4_RNG											
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				RUN4_MAX											
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1

9.2.5.22.4 Fields

Field	Function
31-28 —	Reserved. Always 0.
27-16 RUN4_RNG	Run Length 4 Range. The number of runs of length 4 (for both 0 and 1) detected during entropy generation must be greater than RUN4_MAX - RUN4_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-12 —	Reserved. Always 0.

Table continues on the next page...

Field	Function
11-0 RUN4_MAX	Run Length 4 Maximum Limit. Defines the maximum allowable runs of length 4 (for both 0 and 1) detected during entropy generation. The number of runs of length 4 detected during entropy generation must be less than RUN4_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.23 Statistical Check Run Length 5 Count Register (SCR5C)

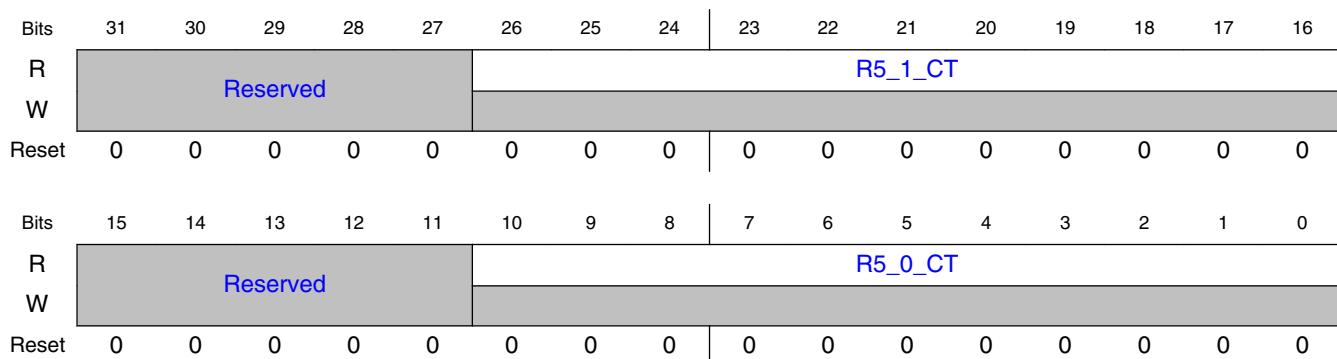
9.2.5.23.1 Offset

Register	Offset	Description
SCR5C	34h	Accessible at this address when MCTL[PRGM] = 0]

9.2.5.23.2 Function

The Statistical Check Run Length 5 Counters Register is a read-only register used to read the final Run Length 5 counts after entropy generation. These counters start with the value in SCR5L[RUN5_MAX]. The R5_1_CT decrements each time five consecutive ones are sampled (preceded by a zero and followed by a zero). The R5_0_CT decrements each time five consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x34) is used as SCR5L if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR5C readback register, as described here.

9.2.5.23.3 Diagram



9.2.5.23.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 R5_1_CT	Runs of One, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.
15-11 —	Reserved. Always 0.
10-0 R5_0_CT	Runs of Zero, Length 5 Count. Reads the final Runs of Ones, length 5 count after entropy generation. Requires MCTL[PRGM] = 0.

9.2.5.24 Statistical Check Run Length 5 Limit Register (SCR5L)

9.2.5.24.1 Offset

Register	Offset	Description
SCR5L	34h	Accessible at this address when MCTL[PRGM] = 1]

9.2.5.24.2 Function

The Statistical Check Run Length 5 Limit Register defines the allowable maximum and minimum number of runs of length 5 detected during entropy generation. To pass the test, the number of runs of length 5 (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 5 must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this address (0x34) is used as SCR5L only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this address is used as SCR5C readback register.

9.2.5.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				RUN5_RNG											
W	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				RUN5_MAX											
W	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1

9.2.5.24.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 RUN5_RNG	Run Length 5 Range. The number of runs of length 5 (for both 0 and 1) detected during entropy generation must be greater than RUN5_MAX - RUN5_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-11 —	Reserved. Always 0.
10-0 RUN5_MAX	Run Length 5 Maximum Limit. Defines the maximum allowable runs of length 5 (for both 0 and 1) detected during entropy generation. The number of runs of length 5 detected during entropy generation must be less than RUN5_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.25 Statistical Check Run Length 6+ Count Register (SCR6PC)

9.2.5.25.1 Offset

Register	Offset	Description
SCR6PC	38h	Accessible at this address when MCTL[PRGM] = 0]

9.2.5.25.2 Function

The Statistical Check Run Length 6+ Counters Register is a read-only register used to read the final Run Length 6+ counts after entropy generation. These counters start with the value in SCR6PL[RUN6P_MAX]. The R6P_1_CT decrements each time six or more

consecutive ones are sampled (preceded by a zero and followed by a zero). The R6P_0_CT decrements each time six or more consecutive zeroes are sampled (preceded by a one and followed by a one). Note that this offset (0x38) is used as SCR6PL if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register, as described here.

9.2.5.25.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved					R6P_1_CT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					R6P_0_CT										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.25.4 Fields

Field	Function
31-27	Reserved. Always 0.
—	
26-16 R6P_1_CT	Runs of One, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.
15-11	Reserved. Always 0.
—	
10-0 R6P_0_CT	Runs of Zero, Length 6+ Count. Reads the final Runs of Ones, length 6+ count after entropy generation. Requires MCTL[PRGM] = 0.

9.2.5.26 Statistical Check Run Length 6+ Limit Register (SCR6PL)

9.2.5.26.1 Offset

Register	Offset	Description
SCR6PL	38h	Accessible at this address when MCTL[PRGM] = 1

9.2.5.26.2 Function

The Statistical Check Run Length 6+ Limit Register defines the allowable maximum and minimum number of runs of length 6 or more detected during entropy generation. To pass the test, the number of runs of length 6 or more (for samples of both 0 and 1) must be less than the programmed maximum value, and the number of runs of length 6 or more must be greater than (maximum - range). If this test fails, the Retry Counter in SCMISC will be decremented, and a retry will occur if the Retry Count has not reached zero. If the Retry Count has reached zero, an error will be generated. Note that this offset (0x38) is used as SCR6PL only if MCTL[PRGM] is 1. If MCTL[PRGM] is 0, this offset is used as SCR6PC readback register.

9.2.5.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								RUN6P_RNG							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RUN6P_MAX							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1

9.2.5.26.4 Fields

Field	Function
31-27 —	Reserved. Always 0.
26-16 RUN6P_RNG	Run Length 6+ Range. The number of runs of length 6 or more (for both 0 and 1) detected during entropy generation must be greater than RUN6P_MAX - RUN6P_RNG, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.
15-11 —	Reserved. Always 0.
10-0 RUN6P_MAX	Run Length 6+ Maximum Limit. Defines the maximum allowable runs of length 6 or more (for both 0 and 1) detected during entropy generation. The number of runs of length 6 or more detected during entropy generation must be less than RUN6P_MAX, else a retry or error will occur. This register is cleared to the default POR value by writing the MCTL[RST_DEF] bit to 1.

9.2.5.27 Status Register (STATUS)

9.2.5.27.1 Offset

Register	Offset
STATUS	3Ch

9.2.5.27.2 Function

Various statistical tests are run as a normal part of the TRNG's entropy generation process. The least-significant 16 bits of the STATUS register reflect the result of each of these tests. The status of these bits will be valid when the TRNG has finished its entropy generation process. Software can determine when this occurs by polling the ENT_VAL bit in the Miscellaneous Control Register.

Note that there is a very small probability that a statistical test will fail even though the TRNG is operating properly. If this happens the TRNG will automatically retry the entire entropy generation process, including running all the statistical tests. The value in RETRY_CT is decremented each time an entropy generation retry occurs. If a statistical check fails when the retry count is nonzero, a retry is initiated. But if a statistical check fails when the retry count is zero, an error is generated by the RNG. By default RETRY_CT is initialized to 1, but software can increase the retry count by writing to the RTY_CT field in the SCMISC register.

All 0s will be returned if this register address is read while the RNG is in Program Mode (see PRGM field in MCTL register). If this register is read while the RNG is in Run Mode the value returned will be formatted as follows.

9.2.5.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																RETRY_CT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFMB	TFP	TFL	TFS	TF6PBR1	TF6PBR0	TF5BR1	TF5BR0	TF4BR1	TF4BR0	TF3BR1	TF3BR0	TF2BR1	TF2BR0	TF1BR1	TF1BR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.27.4 Fields

Field	Function
31-20 —	Reserved. Always 0.
19-16 RETRY_CT	RETRY COUNT. This represents the current number of entropy generation retries left before a statistical test failure will cause the RNG to generate an error condition.
15 TFMB	Test Fail, Mono Bit. If TFMB=1, the Mono Bit Test has failed.
14 TFP	Test Fail, Poker. If TFP=1, the Poker Test has failed.
13 TFLR	Test Fail, Long Run. If TFLR=1, the Long Run Test has failed.
12 TFSB	Test Fail, Sparse Bit. If TFSB=1, the Sparse Bit Test has failed.
11 TF6PBR1	Test Fail, 6 Plus Bit Run, Sampling 1s. If TF6PBR1=1, the 6 Plus Bit Run, Sampling 1s Test has failed.
10 TF6PBR0	Test Fail, 6 Plus Bit Run, Sampling 0s. If TF6PBR0=1, the 6 Plus Bit Run, Sampling 0s Test has failed.
9 TF5BR1	Test Fail, 5-Bit Run, Sampling 1s. If TF5BR1=1, the 5-Bit Run, Sampling 1s Test has failed.
8 TF5BR0	Test Fail, 5-Bit Run, Sampling 0s. If TF5BR0=1, the 5-Bit Run, Sampling 0s Test has failed.
7 TF4BR1	Test Fail, 4-Bit Run, Sampling 1s. If TF4BR1=1, the 4-Bit Run, Sampling 1s Test has failed.
6 TF4BR0	Test Fail, 4-Bit Run, Sampling 0s. If TF4BR0=1, the 4-Bit Run, Sampling 0s Test has failed.
5 TF3BR1	Test Fail, 3-Bit Run, Sampling 1s. If TF3BR1=1, the 3-Bit Run, Sampling 1s Test has failed.
4 TF3BR0	Test Fail, 3-Bit Run, Sampling 0s. If TF3BR0=1, the 3-Bit Run, Sampling 0s Test has failed.
3 TF2BR1	Test Fail, 2-Bit Run, Sampling 1s. If TF2BR1=1, the 2-Bit Run, Sampling 1s Test has failed.
2 TF2BR0	Test Fail, 2-Bit Run, Sampling 0s. If TF2BR0=1, the 2-Bit Run, Sampling 0s Test has failed.
1 TF1BR1	Test Fail, 1-Bit Run, Sampling 1s. If TF1BR1=1, the 1-Bit Run, Sampling 1s Test has failed.
0 TF1BR0	Test Fail, 1-Bit Run, Sampling 0s. If TF1BR0=1, the 1-Bit Run, Sampling 0s Test has failed.

9.2.5.28 Entropy Read Register (ENT0 - ENT15)

9.2.5.28.1 Offset

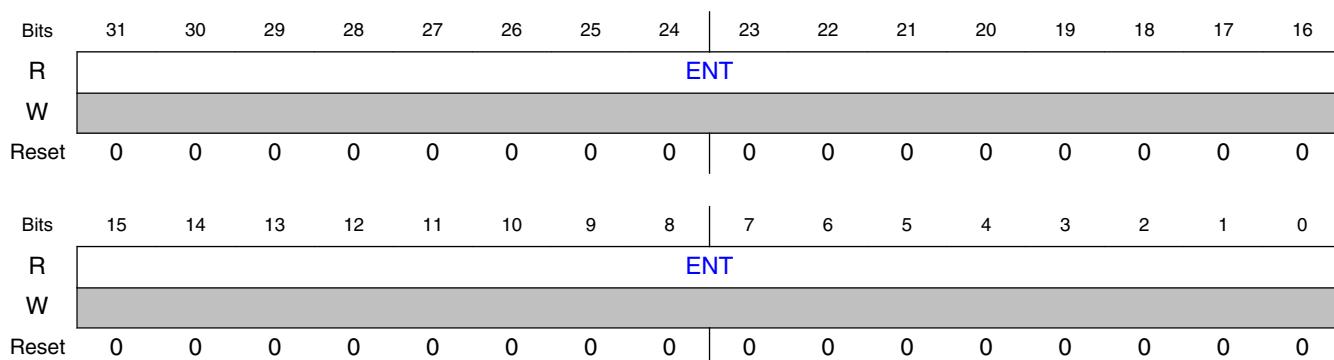
For $a = 0$ to 15:

Register	Offset	Description
ENTa	40h + ($a \times 4h$)	Word a

9.2.5.28.2 Function

The RNG TRNG can be programmed to generate an entropy value that is readable via the SkyBlue bus. To do this, set the MCTL[TRNG_ACC] bit to 1. Once the entropy value has been generated, the MCTL[ENT_VAL] bit will be set to 1. At this point, ENT0 through ENT15 may be read to retrieve the 512-bit entropy value. Note that once ENT15 is read, the entropy value will be cleared and a new value will begin generation, so it is important that ENT15 be read last. These registers are readable only when MCTL[PRGM] = 0 (Run Mode), MCTL[TRNG_ACC] = 1 (TRNG access mode) and MCTL[ENT_VAL] = 1. After at most one (1) bus clock cycle of reading a valid ENT15 register value, reading any ENT0 through ENT15 register would return zeros.

9.2.5.28.3 Diagram



9.2.5.28.4 Fields

Field	Function
31-0 ENT	Entropy Value. Will be non-zero only if MCTL[PRGM] = 0 Run Mode) and MCTL[ENT_VAL] = 1 (Entropy Valid). The most significant bits of the entropy are read from the lowest offset, and the least significant bits are read from the highest offset. Note that reading the highest offset also clears the entire entropy value, and starts a new entropy generation.

9.2.5.29 Statistical Check Poker Count 1 and 0 Register (PKRCNT10)

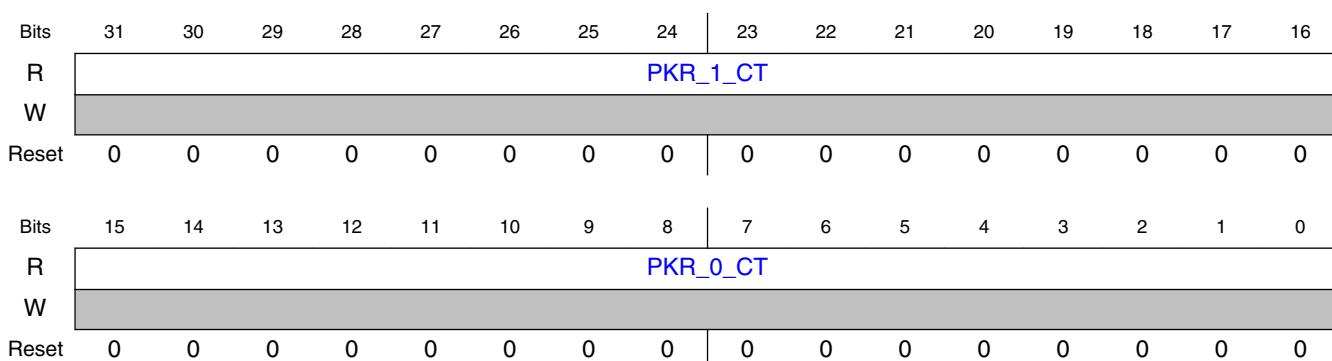
9.2.5.29.1 Offset

Register	Offset
PKRCNT10	80h

9.2.5.29.2 Function

The Statistical Check Poker Count 1 and 0 Register is a read-only register used to read the final Poker test counts of 1h and 0h patterns. The Poker 0h Count increments each time a nibble of sample data is found to be 0h. The Poker 1h Count increments each time a nibble of sample data is found to be 1h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.29.3 Diagram



9.2.5.29.4 Fields

Field	Function
31-16 PKR_1_CT	Poker 1h Count. Total number of nibbles of sample data which were found to be 1h. Requires MCTL[PRGM] = 0.
15-0 PKR_0_CT	Poker 0h Count. Total number of nibbles of sample data which were found to be 0h. Requires MCTL[PRGM] = 0.

9.2.5.30 Statistical Check Poker Count 3 and 2 Register (PKRCNT32)

9.2.5.30.1 Offset

Register	Offset
PKRCNT32	84h

9.2.5.30.2 Function

The Statistical Check Poker Count 3 and 2 Register is a read-only register used to read the final Poker test counts of 3h and 2h patterns. The Poker 2h Count increments each time a nibble of sample data is found to be 2h. The Poker 3h Count increments each time a nibble of sample data is found to be 3h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_3_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_2_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.30.4 Fields

Field	Function
31-16 PKR_3_CT	Poker 3h Count. Total number of nibbles of sample data which were found to be 3h. Requires MCTL[PRGM] = 0.
15-0 PKR_2_CT	Poker 2h Count. Total number of nibbles of sample data which were found to be 2h. Requires MCTL[PRGM] = 0.

9.2.5.31 Statistical Check Poker Count 5 and 4 Register (PKRCNT54)

9.2.5.31.1 Offset

Register	Offset
PKRCNT54	88h

9.2.5.31.2 Function

The Statistical Check Poker Count 5 and 4 Register is a read-only register used to read the final Poker test counts of 5h and 4h patterns. The Poker 4h Count increments each time a nibble of sample data is found to be 4h. The Poker 5h Count increments each time a nibble of sample data is found to be 5h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_5_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_4_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.31.4 Fields

Field	Function
31-16 PKR_5_CT	Poker 5h Count. Total number of nibbles of sample data which were found to be 5h. Requires MCTL[PRGM] = 0.
15-0 PKR_4_CT	Poker 4h Count. Total number of nibbles of sample data which were found to be 4h. Requires MCTL[PRGM] = 0.

9.2.5.32 Statistical Check Poker Count 7 and 6 Register (PKRCNT76)

9.2.5.32.1 Offset

Register	Offset
PKRCNT76	8Ch

9.2.5.32.2 Function

The Statistical Check Poker Count 7 and 6 Register is a read-only register used to read the final Poker test counts of 7h and 6h patterns. The Poker 6h Count increments each time a nibble of sample data is found to be 6h. The Poker 7h Count increments each time a nibble of sample data is found to be 7h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.32.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_7_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_6_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.32.4 Fields

Field	Function
31-16 PKR_7_CT	Poker 7h Count. Total number of nibbles of sample data which were found to be 7h. Requires MCTL[PRGM] = 0.
15-0 PKR_6_CT	Poker 6h Count. Total number of nibbles of sample data which were found to be 6h. Requires MCTL[PRGM] = 0.

9.2.5.33 Statistical Check Poker Count 9 and 8 Register (PKRCNT98)

9.2.5.33.1 Offset

Register	Offset
PKRCNT98	90h

9.2.5.33.2 Function

The Statistical Check Poker Count 9 and 8 Register is a read-only register used to read the final Poker test counts of 9h and 8h patterns. The Poker 8h Count increments each time a nibble of sample data is found to be 8h. The Poker 9h Count increments each time a nibble of sample data is found to be 9h. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.33.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PKR_9_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKR_8_CT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.2.5.33.4 Fields

Field	Function
31-16 PKR_9_CT	Poker 9h Count. Total number of nibbles of sample data which were found to be 9h. Requires MCTL[PRGM] = 0.
15-0 PKR_8_CT	Poker 8h Count. Total number of nibbles of sample data which were found to be 8h. Requires MCTL[PRGM] = 0.

9.2.5.34 Statistical Check Poker Count B and A Register (PKRCNTBA)

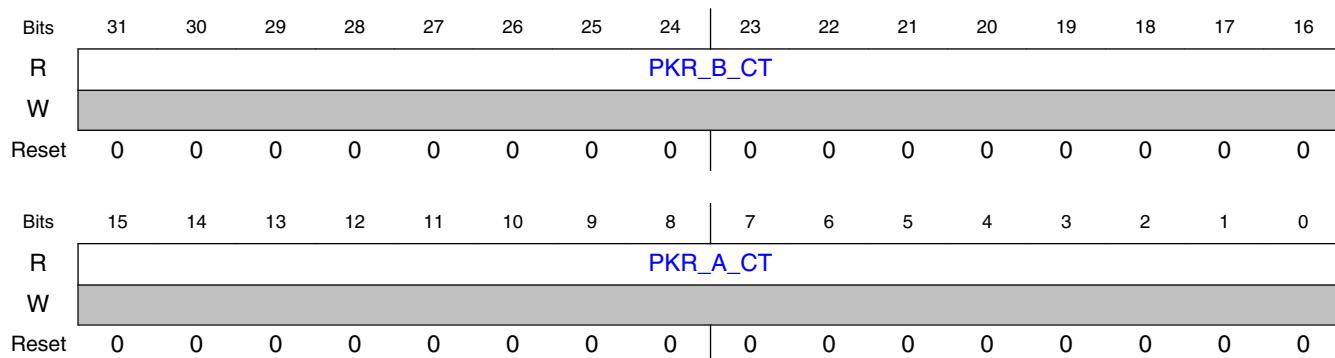
9.2.5.34.1 Offset

Register	Offset
PKRCNTBA	94h

9.2.5.34.2 Function

The Statistical Check Poker Count B and A Register is a read-only register used to read the final Poker test counts of Bh and Ah patterns. The Poker Ah Count increments each time a nibble of sample data is found to be Ah. The Poker Bh Count increments each time a nibble of sample data is found to be Bh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.34.3 Diagram



9.2.5.34.4 Fields

Field	Function
31-16 PKR_B_CT	Poker Bh Count. Total number of nibbles of sample data which were found to be Bh. Requires MCTL[PRGM] = 0.
15-0 PKR_A_CT	Poker Ah Count. Total number of nibbles of sample data which were found to be Ah. Requires MCTL[PRGM] = 0.

9.2.5.35 Statistical Check Poker Count D and C Register (PKRC NTDC)

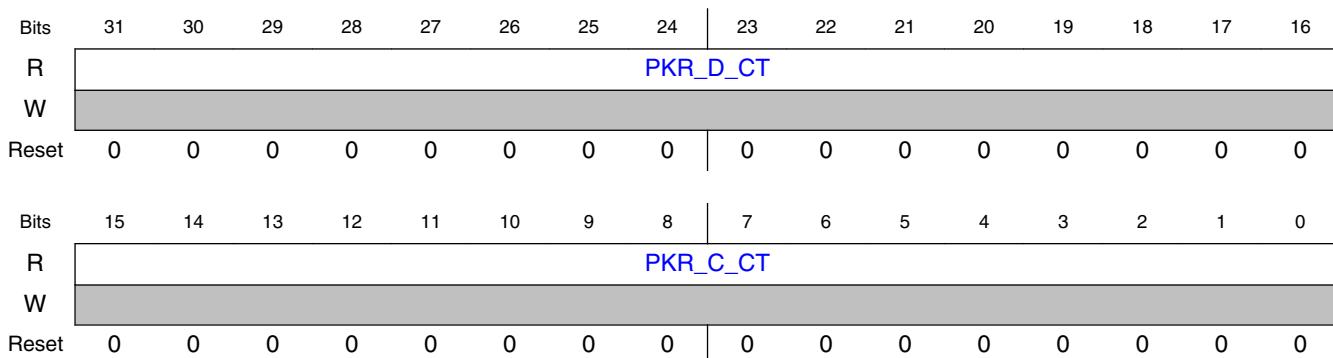
9.2.5.35.1 Offset

Register	Offset
PKRCNTDC	98h

9.2.5.35.2 Function

The Statistical Check Poker Count D and C Register is a read-only register used to read the final Poker test counts of Dh and Ch patterns. The Poker Ch Count increments each time a nibble of sample data is found to be Ch. The Poker Dh Count increments each time a nibble of sample data is found to be Dh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.35.3 Diagram



9.2.5.35.4 Fields

Field	Function
31-16 PKR_D_CT	Poker Dh Count. Total number of nibbles of sample data which were found to be Dh. Requires MCTL[PRGM] = 0.
15-0 PKR_C_CT	Poker Ch Count. Total number of nibbles of sample data which were found to be Ch. Requires MCTL[PRGM] = 0.

9.2.5.36 Statistical Check Poker Count F and E Register (PKRCNTFE)

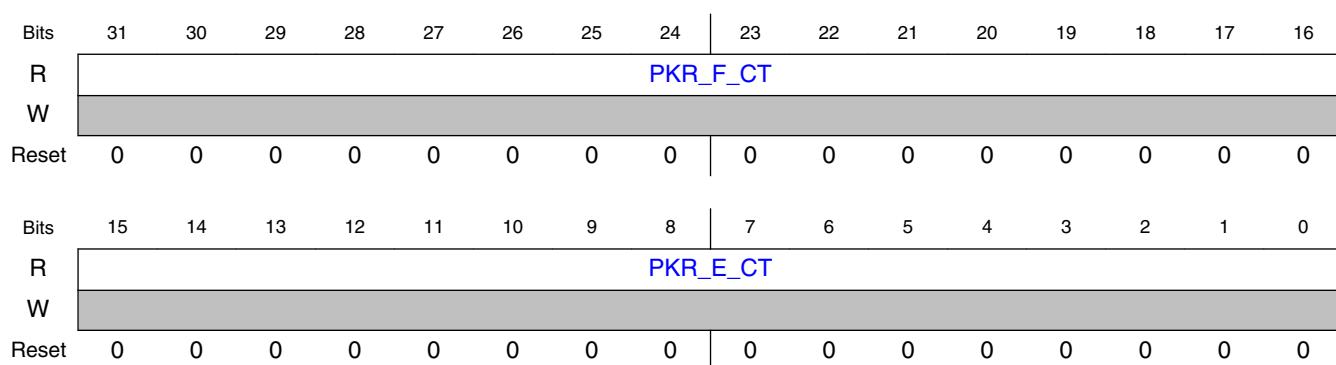
9.2.5.36.1 Offset

Register	Offset
PKRCNTFE	9Ch

9.2.5.36.2 Function

The Statistical Check Poker Count F and E Register is a read-only register used to read the final Poker test counts of Fh and Eh patterns. The Poker Eh Count increments each time a nibble of sample data is found to be Eh. The Poker Fh Count increments each time a nibble of sample data is found to be Fh. Note that this register is readable only if MCTL[PRGM] is 0, otherwise zeroes will be read.

9.2.5.36.3 Diagram



9.2.5.36.4 Fields

Field	Function
31-16 PKR_F_CT	Poker Fh Count. Total number of nibbles of sample data which were found to be Fh. Requires MCTL[PRGM] = 0.
15-0 PKR_E_CT	Poker Eh Count. Total number of nibbles of sample data which were found to be Eh. Requires MCTL[PRGM] = 0.

9.2.5.37 Security Configuration Register (SEC_CFG)

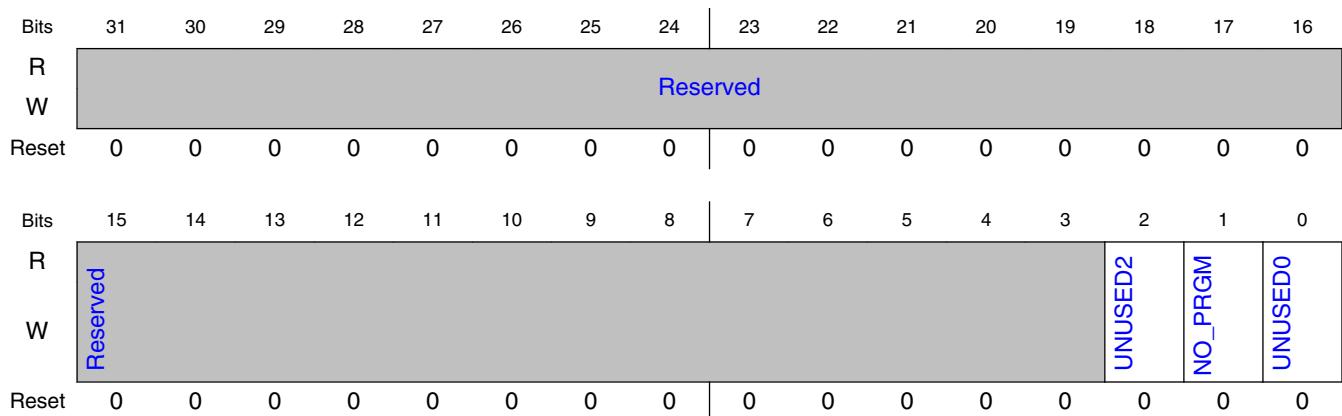
9.2.5.37.1 Offset

Register	Offset
SEC_CFG	A0h

9.2.5.37.2 Function

The Security Configuration Register is a read/write register used to control the test mode, programmability and state modes of the TRNG. Many bits are place holders for this version. More configurability will be added here. Clears on asynchronous system hard reset or POR.

9.2.5.37.3 Diagram



9.2.5.37.4 Fields

Field	Function
31-3 —	Reserved.
2 UNUSED2	This bit is unused. Ignore.
1 NO_PRGM	If set, the TRNG registers cannot be programmed. That is, regardless of the TRNG access mode in the TRNG Miscellaneous Control Register. 0b - Programability of registers controlled only by the Miscellaneous Control Register's access mode bit. 1b - Overrides Miscellaneous Control Register access mode and prevents TRNG register programming.
0 UNUSED0	This bit is unused. Ignore.

9.2.5.38 Interrupt Control Register (INT_CTRL)

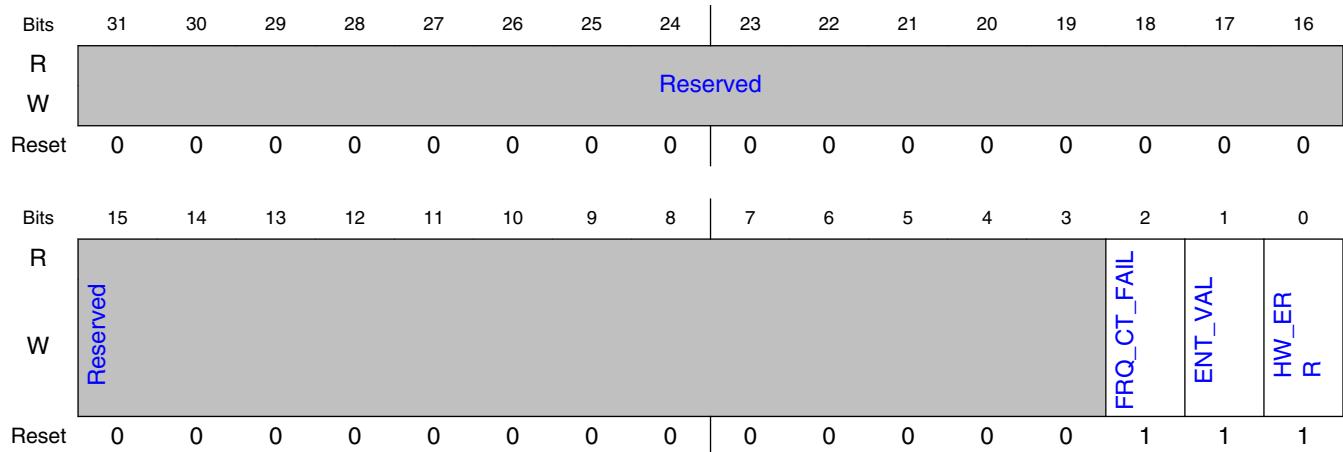
9.2.5.38.1 Offset

Register	Offset
INT_CTRL	A4h

9.2.5.38.2 Function

The Interrupt Control Register is a read/write register used to control the status for the (currently) three important interrupts that are generated by the TRNG. See [INT_STATUS](#) register description. Each interrupt can be cleared by de-asserting the corresponding bit in the [INT_CTRL](#) register. Only a new interrupt will reassert the corresponding bit in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the [MCTL](#) register.

9.2.5.38.3 Diagram



9.2.5.38.4 Fields

Field	Function
31-3	Reserved. Always zero.
—	
2	Same behavior as bit 0 of this register.
FRQ_CT_FAIL	0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
1	Same behavior as bit 0 of this register.
ENT_VAL	0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
0	Bit position that can be cleared if corresponding bit of INT_STATUS register has been asserted.
HW_ERR	0b - Corresponding bit of INT_STATUS register cleared. 1b - Corresponding bit of INT_STATUS register active.

9.2.5.39 Mask Register (INT_MASK)

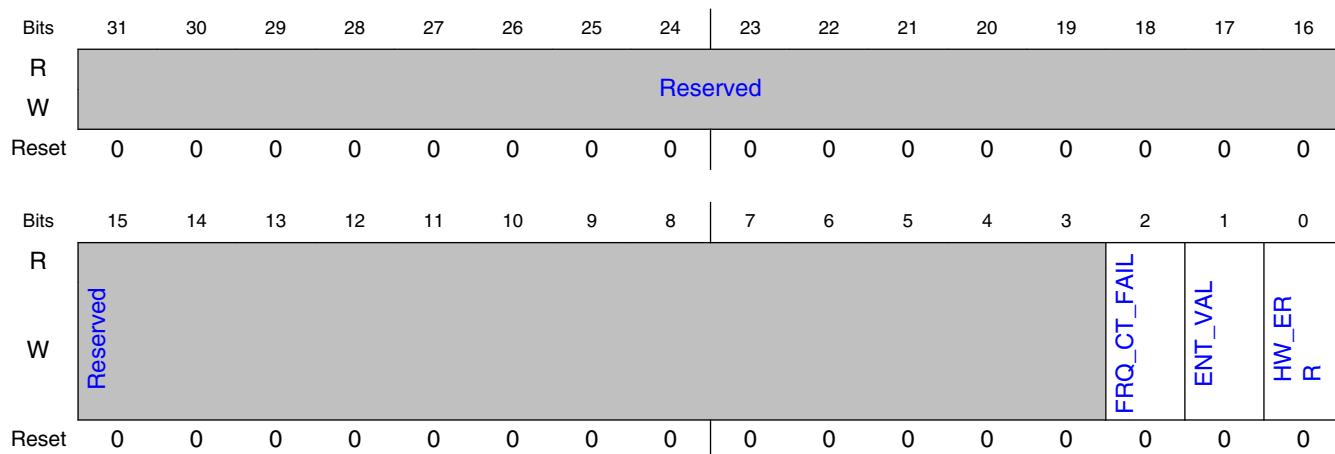
9.2.5.39.1 Offset

Register	Offset
INT_MASK	A8h

9.2.5.39.2 Function

The Interrupt Mask Register is a read/write register used to disable/mask the status reporting of the (currently) three important interrupts that are generated by the TRNG. See [INT_STATUS](#) register description. Each interrupt can be masked/disabled by de-asserting the corresponding bit in the [INT_MASK](#) register. Only setting this bit high will re-enable the interrupt in the status register. Even if the interrupt is cleared or masked, interrupt status information can be read from the [MCTL](#) register.

9.2.5.39.3 Diagram



9.2.5.39.4 Fields

Field	Function
31-3	Reserved.
—	
2	Same behavior as bit 0 of this register.
FRQ_CT_FAIL	0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
1	Same behavior as bit 0 of this register.
ENT_VAL	0b - Same behavior as bit 0 of this register. 1b - Same behavior as bit 0 of this register.
0	Bit position that can be cleared if corresponding bit of INT_STATUS has been asserted.
HW_ERR	0b - Corresponding interrupt of INT_STATUS is masked. 1b - Corresponding bit of INT_STATUS is active.

9.2.5.40 Interrupt Status Register (INT_STATUS)

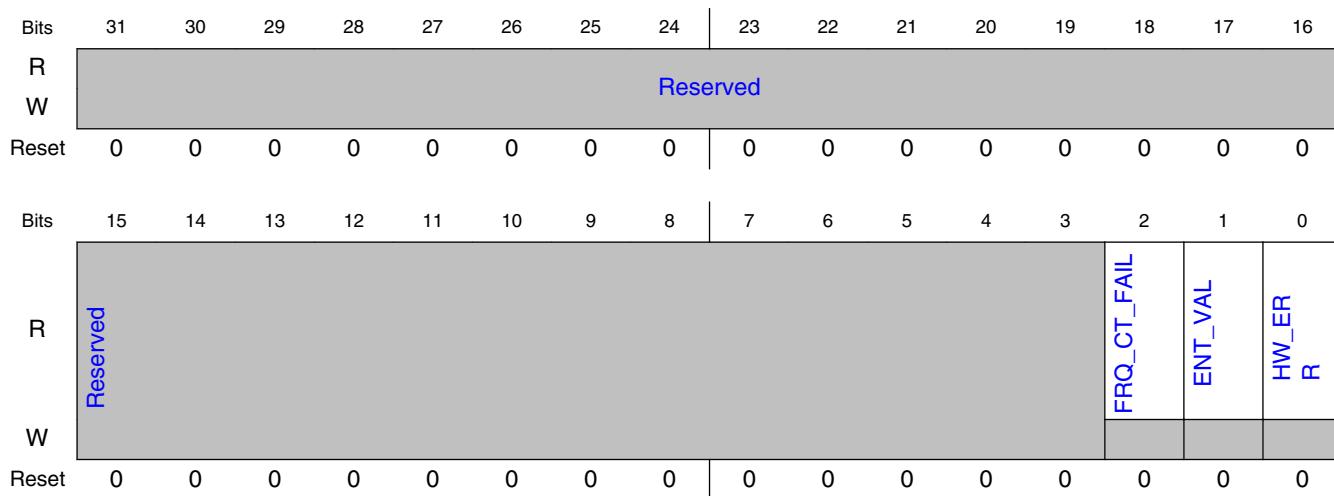
9.2.5.40.1 Offset

Register	Offset
INT_STATUS	ACh

9.2.5.40.2 Function

The Interrupt Status Register is a read register used to control and provide status for the (currently) three important interrupts that are generated by the TRNG. The TRNG interrupt signal alerts that TRNG has either generated a Frequency Count Fail, Entropy Valid or Error Interrupt. The cause of the interrupt can be decoded by checking the least significant bits of the INT_STATUS register. Each interrupt can be temporarily cleared by de-asserting the corresponding bit in the [INT_CTRL](#) register. To mask the interrupts, clear the corresponding bits in the [INT_MASK](#) register. The description of each of the 3 interrupts is defined in the Block Guide under the MCTL register description. Even if the interrupt is cleared or masked, interrupt status information can be read from the MCTL register.

9.2.5.40.3 Diagram



9.2.5.40.4 Fields

Field	Function
31-3	Reserved.

Table continues on the next page...

Field	Function
—	
2 FRQ_CT_FAIL	Read only: Frequency Count Fail. The frequency counter has detected a failure. This may be due to improper programming of the FRQMAX and/or FRQMIN registers, or a hardware failure in the ring oscillator. 0b - No hardware nor self test frequency errors. 1b - The frequency counter has detected a failure.
1 ENT_VAL	Read only: Entropy Valid. Will assert only if TRNG ACC bit is set, and then after an entropy value is generated. Will be cleared when ENT15 is read. (ENT0 through ENT14 should be read before reading ENT15). 0b - Busy generation entropy. Any value read is invalid. 1b - TRNG can be stopped and entropy is valid if read.
0 HW_ERR	Read: Error status. 1 = error detected. 0 = no error. Any HW error in the TRNG will trigger this interrupt. 0b - no error 1b - error detected.

9.2.5.41 Version ID Register (MS) (VID1)

9.2.5.41.1 Offset

Register	Offset
VID1	F0h

9.2.5.41.2 Function

The Version ID Register (VID1) is a MSW read only register used to identify the version of the TRNG in use. This register as well as [VID2](#) should both be read to verify the expected version.

9.2.5.41.3 Diagram

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

9.2.5.41.4 Fields

Field	Function
31-16 IP_ID	Shows the IP ID. 0000000000110000b - ID for TRNG.
15-8 MAJ_REV	Shows the IP's Major revision of the TRNG. 00000001b - Major revision number for TRNG.
7-0 MIN_REV	Shows the IP's Minor revision of the TRNG. 0000000b - Minor revision number for TRNG.

9.2.5.42 Version ID Register (LS) (VID2)

9.2.5.42.1 Offset

Register	Offset
VID2	F4h

9.2.5.42.2 Function

The Version ID Register LSB is a read only register used to identify the architecture of the TRNG in use. This register as well as VID1 should both be read to verify the expected version.

9.2.5.42.3 Diagram

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

9.2.5.42.4 Fields

Field	Function
31-24	Shows the compile options for the TRNG.
ERA	00000000b - COMPILE_OPT for TRNG.
23-16	Shows the integration options for the TRNG.
INTG_OPT	00000000b - INTG_OPT for TRNG.
15-8	Shows the IP's ECO revision of the TRNG.
ECO_REV	00000000b - TRNG_ECO_REV for TRNG.
7-0	Shows the IP's Configuration options for the TRNG.
CONFIG_OPT	00000000b - TRNG_CONFIG_OPT for TRNG.

Chapter 10

Secure Non-Volatile Storage (SNVS)

10.1 Chip-specific SNVS information

NOTE

Tamper Detection feature and ZMK hardware programming are not applicable on this device.

SNVS HP security violation inputs:

- sec_vio_in_0 - disabled
- sec_vio_in_1 - JTAG active
- sec_vio_in_2 - WDOG2 reset
- sec_vio_in_3 - disabled
- sec_vio_in_4_b - disabled
- sec_vio_in_5_b - GPIO_EMCA_19

NOTE

For power saving, register access to SNVS_LP domain could be very slow, due to the low speed (XTAL32K) clocking structure of SNVS_LP domain.

10.2 SNVS introduction

SNVS is a companion module to the DCP module and serves as the SOC's central reporting point for security-relevant events such as the success or failure of boot software validation and the detection of security threat events. This security event information determines whether the SoC (hardware and software) is in the proper state to allow the DCP to use persistent and ephemeral secrets. Based on configuration fuses and configured bits within registers, SNVS is able to detect a variety of security violation inputs and perform the configured policy enforcement actions.

SNVS incorporates both security and non-security functionality.

SNVS security functions:

- Monitor security events in the SoC, and respond as per configured security policy
- Protect certain security-critical data objects
 - Master Key - can be used by DCP when encrypting or decrypting blobs
 - One-Time Programmable Master Key (OTPMK) from fuses can be used as the source of the Master Key
 - Zeroizable Master Key (ZMK) - can be used as the source of the Master Key
 - OTPMK XOR ZMK (Combined Master Key, CMK) - can be used as the source of the Master Key
 - Monotonic Counter (MC) - a monotonically increasing counter that can be used for replay detection
 - Secure Realtime Counter (SRTC) - realtime counter that can't be altered by untrusted software
- Preserve state of the data objects listed above (if the SNVS_LP power input is connected to an uninterrupted power supply, e.g. a coin cell battery)
 - ZMK value maintained when main SoC is powered off
 - MC value maintained when main SoC is powered off
 - SRTC continues to count when main SoC is powered off

SNVS non-security functions:

- Realtime Counter (RTC) - a software accessible realtime counter
 - RTC can be set to the value in the SRTC
- Periodic Interrupt - a hardware-generated interrupt that occurs periodically at software-specified frequency
- General Purpose Register - a set of registers used to hold 128 bits of data specified by software
 - If the SNVS_LP power input is connected to an uninterrupted power supply, e.g. a coin cell battery, the GPR value is maintained when main SoC is powered off
- Chip power-on/power-off - If the SNVS_LP power input is connected to an uninterrupted power supply and the Power On button input signal is connected to a power button external to the chip, logic within SNVS_LP can be used to wake the chip from a power down.

10.2.1 SNVS feature list

The following table summarizes the features of SNVS:

Table 10-1. SNVS feature list

Feature	Description	Links for Further Information
Security State Machine (SSM)	<ul style="list-style-type: none"> • Receives configuration inputs from SoC fuses • Receives security violation inputs from the various detectors in the chip • Tracks security state • Generates security state outputs to DCP and other logic within the chip • Can request system hard reset in case of non-recoverable violation • Programmable high assurance counter (HAC) to control time delay before system hard reset request generation 	Security state machine
Master key checking and control	<ul style="list-style-type: none"> • Performs validity checks for the one-time programmable master key before allowing DCP to use it • Performs validity checks for the zeroizable master key before allowing DCP to use it • Selects the device-specific master key value as the OTPMK, the ZMK, or the bit-wise exclusive OR of both the OTPMK and the ZMK 	Configuring Master Key checking and control
Zeroizable master key (ZMK).	<ul style="list-style-type: none"> • The ZMK value can be programmed via software • The ZMK value is zeroized when a security violation occurs • If the SNVS_LP power input is connected to an uninterrupted power supply (see xref href="snvs_power_domains.dita#power_domains"/>, the value is maintained even when the main SoC is powered off. 	Zeroizable master key
Secure real time counter (SRTC)	<ul style="list-style-type: none"> • The SRTC is driven by a dedicated clock that is functionally independent of the chip configuration. • The SRTC does not rollover. Instead the SRTC logic issues an alarm if the SRTC reaches its maximum value. • Programmable time alarm interrupt <ul style="list-style-type: none"> • Software can program the SRTC to generate an interrupt at a specific time. • If the main SoC is powered down at the programmed alarm time and the wake-up external alarm is enabled, the SRTC generates a wake-up alarm via an external pin. (Assuming that the SNVS_LP power input is connected to an uninterrupted power supply, see xref href="snvs_power_domains.dita#power_domains"/>) • The SRTC value is marked as invalid if an SNVS tamper event is detected. 	SNVS_LP Secure Real Time Counter (SRTC)
Real time counter (RTC)	<ul style="list-style-type: none"> • The RTC is driven by a dedicated clock, which is off when the system power is down. • The RTC can be synchronized to the value of the Secure Real Time Counter. • Programmable time alarm interrupt • Periodic interrupt can be generated with software-selected frequency. 	SNVS_HP Real Time Counter
Monotonic counter	<ul style="list-style-type: none"> • The monotonic counter can only increment. • The monotonic counter does not rollover. Instead the monotonic counter logic issues an alarm if the monotonic counter reaches its maximum value. • The monotonic counter value is marked as invalid if an SNVS tamper event is detected. • If the SNVS_LP power input is connected to an uninterrupted power supply (see xref href="snvs_power_domains.dita#power_domains"/>) 	Using the Monotonic Counter (MC)

Table continues on the next page...

Table 10-1. SNVS feature list (continued)

Feature	Description	Links for Further Information
	href="snvs_power_domains.dita#power_domains"/>, the monotonic counter value is retained even if the main chip is powered down.	
General-purpose register	<ul style="list-style-type: none"> The general-purpose register is available to software to store 128 bits of data. The general-purpose register is zeroized when a security violation is detected. If the SNVS_LP power input is connected to an uninterrupted power supply (see xref href="snvs_power_domains.dita#power_domains"/>), the general-purpose register value is retained even if the main chip is powered down. 	Using the General-Purpose Register
Register access restrictions	<ul style="list-style-type: none"> Registers can be programmed only when the SNVS is in functional state, i.e. not in scan mode. Some registers/values can be written only once per boot cycle. 	privileged and non-privileged registers
Violation/tamper detection and reporting	<ul style="list-style-type: none"> Detects (internal to the block) the following security violations: <ul style="list-style-type: none"> Scan exit Power glitch Invalid OTPMK (ECC check failure) ZMK ECC check failure SRTC rollover MC rollover Detects the following security violations: <ul style="list-style-type: none"> Software-reported violations 5 security violation inputs Direct connections to DCP to lock out access to the OTPMK/ZMK and force zeroization of sensitive information Configurably triggers a device hard reset. Configurably reports to software (interrupts) all security violation and functional events 	Security violation policy
Wakeup from power off	<ul style="list-style-type: none"> Input signal from off chip requests SNVS_LP to power on the main SoC (Assuming that the SNVS_LP power input is connected to an uninterrupted power supply (see SNVS power domains)). Hardware debounces the input signal using software-specified signal bounce characteristics 	LP Wake-Up Interrupt Enable

10.2.2 SNVS functional description

The low-power section of SNVS is intended to be initialized at SNVS_LP POR, which normally occurs very rarely (when a new coin cell battery is installed). At this time software initializes the SNVS as described in [Initialization guidelines](#). Once configured, much of the security functionality of SNVS operates automatically. However, there are a few SNVS security functions that involve interaction with software:

- reading the Secure Realtime Counter (SRTC) - this can be read, but not significantly altered¹, by runtime software

1. The SRTC cannot be written after it has been locked by initialization software, but the clock rate can be adjusted until the adjustment bits are also locked.

- reading or updating the Monotonic Counter (MC) - this can be read or incremented by runtime software
- SNVS also stores certain data whose integrity or confidentiality must be protected (e.g. secure real-time clock, monotonic counter, zeroizable master key).

One of the major security functions of SNVS is to sense security-related events on the device and control the device's security state per the software-configured security policy. Once SNVS is initialized, sensing these events and initiating a response is handled by hardware. Certain security-relevant events are detected within SNVS itself (e.g. power on reset, boot from SoC ROM, power glitch), whereas other security-related events are detected via sensors on or off-chip and passed to SNVS via input signals. For instance, these other sensors might detect voltage or temperature or clock frequency out of acceptable ranges.

SNVS implements several non-security features that involve software interaction:

- reading or writing the Realtime Counter (RTC) (This is a non-privileged operation.) - software can also instruct SNVS to load the current SRTC value into the RTC
- reading or writing the General Purpose Register (GPR) (Note that there may be a significant delay when reading or writing registers in the LP section if the LP clock is different from the HP clock.)

The following sections describe in more detail the operation of SNVS.

10.3 SNVS Structure

The SNVS incorporates several features that help to ensure the security of data stored in the device.

- A security state machine that responds to various security conditions
 - hardware security violation inputs
 - hardware tamper inputs
 - software-implemented security checks
- Security-state-machine derived outputs that tell the DCP whether the device state is Trusted, Secure, NonSecure, or Fail
- Automatic zeroization of the Zeroizable Master Key if a tamper event is detected
- Automatic zeroization of the General Purpose Register if a tamper event is detected
- A power glitch detector that alerts the security state machine when a power glitch causes a power glitch detector bit to flip
- Validity flag that indicates if the value in the monotonic counter is valid
- Validity flag that indicates if the value in the secure realtime counter is valid

SNVS is organized as two major sub-modules:

- Low-Power Section of SNVS (SNVS_LP)

The SNVS_LP section provides hardware that enables secure storage and protection of sensitive data. The SNVS module is designed to safely hold security-related data such as cryptographic key, time counter, monotonic counter, and general purpose security information.

The SNVS_LP block implements the following functional units:

- Zeroizable Master Key
- Control and Status Registers
- General Purpose Registers
- Monotonic Counter
- Secure Real Time Counter

When the LP section is powered by a backup battery the state of these registers is maintained even when the main chip power is off. (see [SNVS power domains](#))

- High-Power Section of SNVS (SNVS_HP)

The SNVS_HP section contains all SNVS status and configuration registers. It implements all features that enable system communication and provisioning of the SNVS_LP section. The SNVS_HP section also incorporates the security state machine, which controls the system security state, and the master key control block, which is responsible for checking and selecting the master key value. The SNVS_HP also incorporates the Zeroizable Master Key programming mechanism and the OTPMK logic.

The SNVS_HP provides an interface between SNVS_LP and the rest of the system.

The SNVS_HP block implements the following functional units:

- IP Bus Interface
- SNVS_LP Interface
- Security State Machine (SSM)
- Zeroizable Master Key Programming Mechanism
- Master Key Control block
- Real Time Counter with Alarm Control and Status Registers
- Control and status registers

SNVS_HP is in the chip's power supply domain and thus receives power along with the rest of the chip.

The following figure illustrates the structure of SNVS.

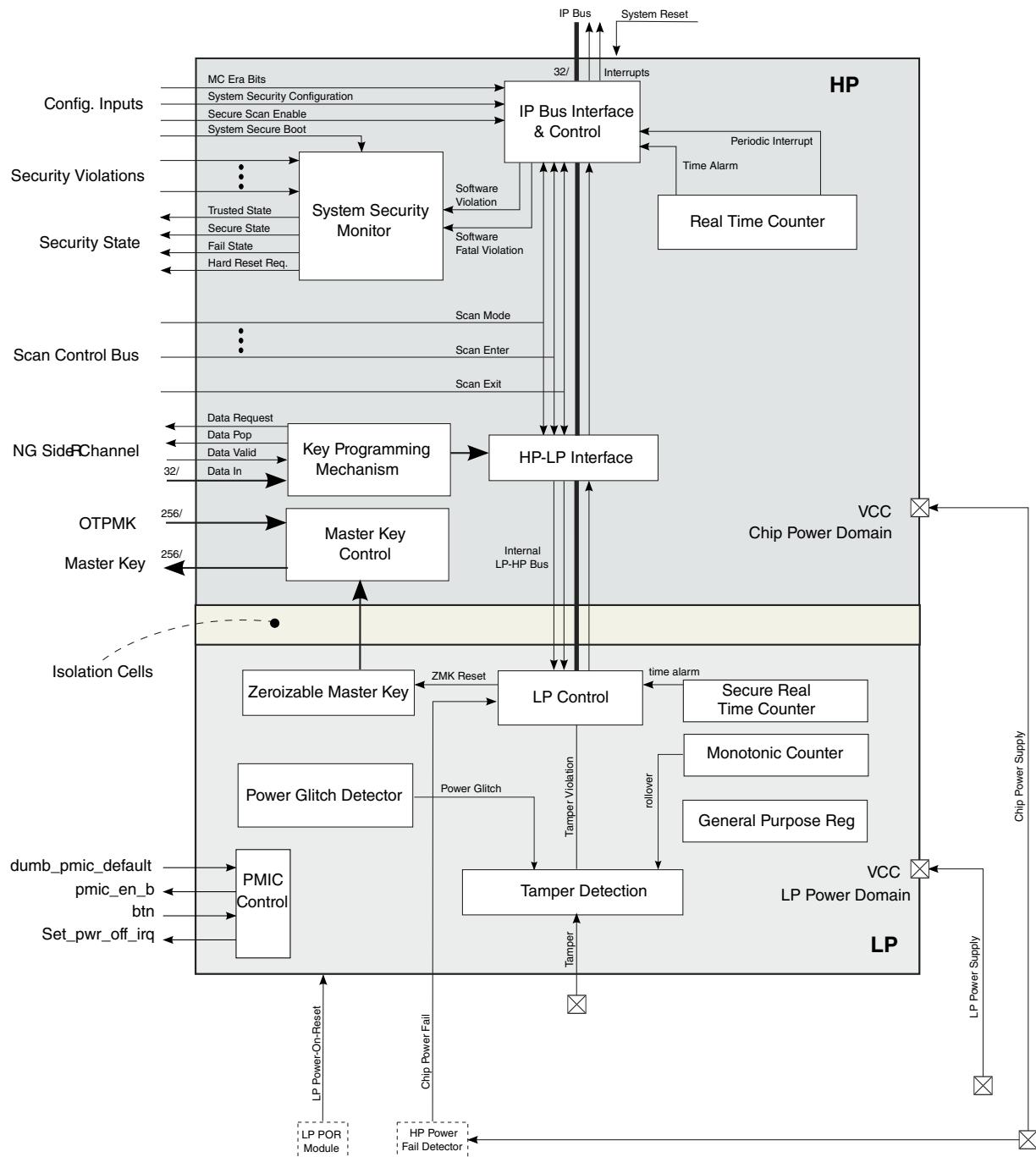


Figure 10-1. SNVS Block Diagram

10.3.1 SNVS power domains

In some versions of SNVS (including this version), the LP (Low Power) section is implemented in an independent power domain from the HP (High Power) section, and most other logic on the chip. Throughout the SNVS documentation whenever mention is

made of "always-on" logic, this assumes a version of SNVS that implements an independent power domain for the LP section, and that the power for this section is supplied by an uninterrupted power supply. The purpose for the independent power domain is so that data can be retained and certain logic can remain functional even when the main chip logic is powered down. But this is possible only if the LP domain remains powered via an uninterrupted power supply when the main chip power domain is powered off. Usually this uninterrupted power supply would be a coin-cell battery, with possibly some power management logic to power the LP section from main power (and perhaps recharge the coin cell battery) when main power is on, and switch to coin-cell power when the main power is off. In versions of SNVS with an independent LP power domain the LP section can be electrically isolated from the rest of the chip logic to ensure that its logic does not get corrupted when the main chip is powered down. If the battery runs down or is removed, an LP POR will occur when the LP section next powers up. Note that some OEMs may choose to connect LP power to HP/main chip power and dispense with a coin cell battery. In that case the SNVS will operate the same as an SNVS without an independent LP power domain. No state will be retained in the LP section when the chip is powered down, and an LP POR will occur whenever there is an HP POR.

10.3.2 SNVS clock sources

The SNVS has the following clock sources:

- System peripheral clock input. This clock is used by the SNVS's internal logic, for example, the Security State Machine. This clock can be gated outside of the module when the SNVS indicates that it is not in use.
- HP RTC clock. This clock is used by SNVS_HP real-time counter. This clock does not need to be synchronous with other clocks.

10.4 Security violation policy

SNVS is intended to detect, and potentially initiate a response to, various security-relevant events in the chip. The nature of the response to various types of security-relevant events can be set by configuring the SNVS security policy appropriately. SNVS detects certain security-relevant events either within its HP section, or via "Security Violation" inputs to the HP section. The following security-relevant events are detected within SNVS_HP hardware:

- OTPMK Hamming code error
- ZMK Hamming code error

The following security-relevant events are detected external to SNVS and are reported to SNVS_HP via "Security Violation" input signals:

- Security Violation 0
- Security Violation 1
- Security Violation 2
- Security Violation 3
- Security Violation 4
- Security Violation 5

Software can also report a security violation to SNVS by writing into the SNVS_HP Command register (HPCOMR).

In addition to the SNVS_HP-detected security-relevant events mentioned, SNVS also detects certain other security-related events in the SNVS_LP section and reacts to them even when main SoC power is off:

- Power Glitch Violation
- SRTC Rollover Violation
- MC Rollover Violation

The SoC firmware specifies SNVS's response to security violations by configuring the SNVS_HP security policy. At boot time the SoC boot firmware also checks the status of the SNVS_LP section. If the LP section has not been securely initialized, or has been tampered, or the coin cell battery has been replaced, the status check will indicate that the state of the LP section is invalid, and the boot firmware will configure the SNVS_LP security policy. The configured security policy determines how security-related events affect the SoC security state, which is tracked by the SNVS Security State Machine as explained in the next section.

10.4.1 Security state machine

SNVS implements a security state machine (SSM) that tracks the security state of the SoC. The states and state transitions of the SSM affect certain security actions taken by SNVS and DCP. For instance, in some security states DCP will use a test key for blob operations, whereas in other security states DCP will use a secret key. Some state transitions cause SNVS and DCP to zeroize certain secret values.

The following figure and table describe the SSM's states and transitions.

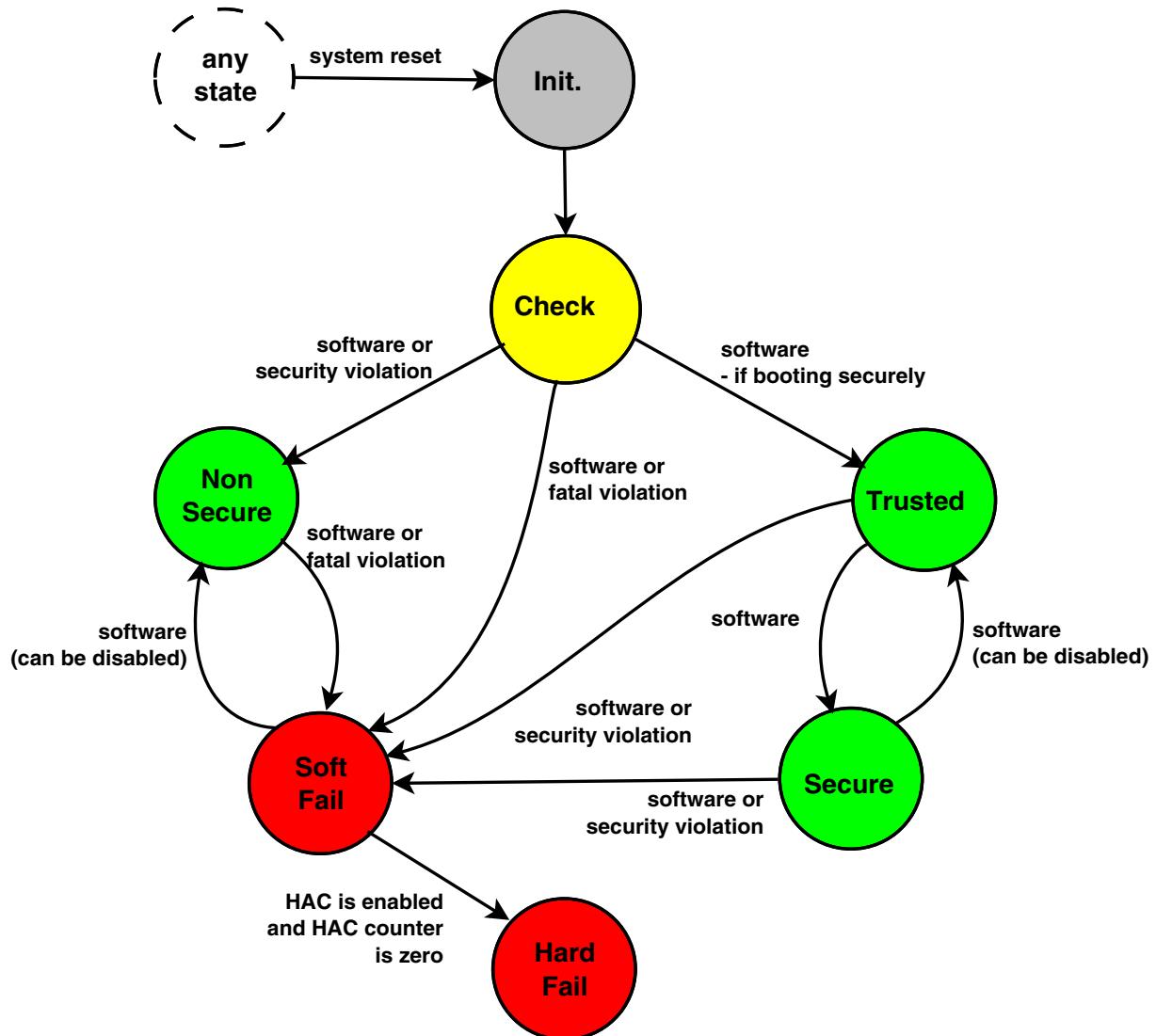


Figure 10-2. Security State Machine

Table 10-2. Security State Machine State Definitions

SSM State	What happens in this state	Security state signals sent to DCP:	Transitions from this SSM State to:
Init <i>(System enters this state at main SoC POR.)</i>	In this state the SSM ignores all security violations sources; security violations are not recorded and not responded to in the SNVS_HP domain. SNVS registers cannot be programmed in this state.	Non-Secure (i.e. not Secure, not Trusted and not Fail)	<ul style="list-style-type: none"> • Check state - if the clocks are stable and fuse values can be read accurately the SSM transitions from Init state to Check state.
Check	System performs the check sequence, which consists of various hardware health checks. While in Check, the SNVS receives various inputs, including	Non-Secure (i.e. not Secure, not Trusted and not Fail)	<ul style="list-style-type: none"> • Trusted state - If software writes a 1 to the control bit in the HP Command Register while the SSM is in Check state, this initiates a transition to Trusted state. The transition is

Table continues on the next page...

**Table 10-2. Security State Machine State Definitions
(continued)**

SSM State	What happens in this state	Security state signals sent to DCP:	Transitions from this SSM State to:
	<p>the fuse values from the SFP, security violation status, and a pass/fail indication from the boot firmware.</p> <p>SNVS registers cannot be programmed in this state, with the exception of several bits in the HP Command Register.</p>		<p>allowed if the system is booting from ROM internal to the chip, or if the System Security Configuration is Fab (000). Upon transition to Trusted state, SNVS_LP is reset if it was previously provisioned in the Non-secure state.</p> <ul style="list-style-type: none"> • Non-secure state - If a non-fatal security violation is detected while in Check state the SSM transitions to the Non-secure state. • Soft Fail state - If a fatal violation is detected while in Check state the SSM transitions to the Soft Fail state.
Non-Secure	In this state, the secure and trusted mode indication signals are not set. In Non-Secure state any write accesses to the SNVS_LP registers are denied if SNVS_LP was provisioned in the Trusted state or Secure state.	Non-Secure (i.e. not Secure, not Trusted and not Fail)	<ul style="list-style-type: none"> • Soft Fail state - If a fatal violation is detected while in Non-Secure state the SSM transitions to the Soft Fail state.
Trusted	<p>In this state, the trusted and secure mode indication signals are asserted. The fact that the SSM is in the Trusted state indicates that there are no hardware security violations and that HAB was satisfied with its security checks (including validating the signature over the next boot software).</p> <p>In this state SNVS signals to DCP indicating that DCP is allowed to use the Trusted/Secure state values of the blob master key, black key encryption keys, and trusted descriptor signing keys. Note that the PRI_BLOB bits in the DCP Security Configuration register affect the derivation of the blob master key only when the SSM is in Trusted state.</p>	Trusted, Secure	<ul style="list-style-type: none"> • Secure state - If software writes to the SSM_ST bit in the HP Command Register while SSM is in the Trusted state, this will trigger a transition to Secure state. • Soft Fail state - If the SSM detects a security violation condition while SSM is in the Trusted state, the SSM immediately (without clock) transitions to the Soft Fail state. If software writes to the HP Command register's SW_SFV or SW_SV bits while the SSM is in the Trusted state, the SSM will transition to Soft Fail state.
Secure	<p>In this state, the secure mode indication signal is asserted and trusted mode indication signal is de-asserted. This indicates that DCP is allowed to use the Trusted/Secure state values of the blob master key, black key encryption keys, and trusted descriptor signing keys. The only difference between Secure and Trusted states is that the derivation of the blob master key is different. The PRI_BLOB bits in the DCP Security Configuration register factor into the blob master key derivation only in</p>	Secure	<ul style="list-style-type: none"> • Trusted state - If software writes to the SSM_ST bit in the HP Command Register while SSM is in the Secure state, this will trigger a transition to Trusted state. (The Secure state to Trusted state transition can be disabled by setting the SSM_ST_DIS bit in the HP Command Register.) • Soft Fail state - If the SSM detects a security violation condition while SSM is in the Trusted state, the SSM immediately (without clock) transitions to the Soft Fail state. If software writes

Table continues on the next page...

**Table 10-2. Security State Machine State Definitions
(continued)**

SSM State	What happens in this state	Security state signals sent to DCP:	Transitions from this SSM State to:
	Trusted state. This is intended to allow HAB and/or secure provisioning software to encrypt/decrypt blobs that cannot be encrypted or decrypted by runtime software.		to the HP Command register's SW_SFV or SW_SV bits while the SSM is in the Trusted state, the SSM will transition to Soft Fail state.
Soft fail	Upon transitioning to Soft Fail the SNVS signals the DCP to use the test master key for blobs and to zeroize the KEKs, TDSK and all key registers. The SNVS also generates an interrupt to inform software of the Soft Fail, and if configured to do so, starts the HAC_Counter countdown toward Hard Fail. The device can operate with the SSM in the Soft Fail state indefinitely; however, no operations involving the DCP can occur unless the SNVS is transitioned from Soft Fail state to the Non-Secure state. Transition from Soft Fail to the Non-Secure state can be triggered by software via a write to the HP Command register SSM_ST bit, unless the HAC counter is actively counting or the transition to the Non-Secure state has been disabled via the SSM_SFNS_DIS bit. But the black keys and trusted descriptors that existed prior to the entry into Soft Fail will not be recoverable even in Non-Secure state and until the SoC is reset, the DCP will be unable to perform blob operations with the secret master key. SNVS registers cannot be programmed in this state, except for several bits in the HP Command Register.	Fail	<ul style="list-style-type: none"> Hard Fail state - If the HP Command Register HAC_EN bit is 1 when the SSM enters the Soft Fail state, the High Assurance Counter is loaded from the High Assurance Counter IV register and then the High Assurance Counter begins counting down. Software can stop this counter by writing to the HAC_STOP bit in the HP Command Register. If software fails to stop the High Assurance Counter before it counts down to zero the SSM transitions to the Hard Fail state. This acts as a "deadman switch" to reset the SoC if the software is hung. Non-Secure state Software can initiate a transition from Soft-Fail state to Non-Secure state by writing a 1 to the SSM_ST bit in the HP Command Register, unless the transition has been disabled by setting the SSM_ST_DIS bit in the HP Command Register or the High Assurance Counter has been activated but not stopped, or a fatal security violation is active. In these cases the SSM stays in the soft fail state.
Hard Fail	Entering this state triggers the hard reset request output, which should be used in the system to perform a hardware reset without the aid of software. SNVS registers cannot be programmed in this state.	Fail	<ul style="list-style-type: none"> Hard-Fail state - Once the SSM enters the Hard-Fail state it remains in the Hard-Fail state until the system is reset.

All HP security violation sources are classified into one of three categories: disabled, non-fatal security, and fatal security. Disabled security violations have no effect on the SSM. Fatal security violations always result in the SSM transition to the Soft Fail state. The non-fatal security violations result in a transition from Check to Non-Secure or from Trusted/Secure states to Soft Fail state.

- Disabled violation - SNVS records the violation does not otherwise react to the violation.
- Non-fatal security violation - SNVS records the violation but does not cause security registers to be cleared
- Fatal security violation - SNVS records the violation and causes security registers to be cleared.

Regardless of the category all security violation events are recorded in the corresponding status registers.

10.4.2 SNVS interrupts, alarms, and security violations

When SNVS detects security violations or tamper events, SNVS responds as dictated by the security policies pre-configured by software. This security policy configuration specifies under what circumstances SNVS asserts the following interrupt and alarm signals:

Table 10-3. Interrupts and alarms summary

Interrupt/violation	Security Event	Default configuration ¹	Configuration options
SNVS security interrupt	Security violation input asserted	Disable	Enable/disable
	SSM transitions to the soft fail state	Enable	-
	LP security violation asserted	Disable	Enable/disable
SNVS security violation	SSM transitions to the soft fail state	Enable	-
SNVS hard failure reset	SSM transitions to the hard fail state	Enable	-

1. Default behavior refers to the setting after Reset

10.4.3 Configuring SNVS's response to a security violation or tamper event

SNVS's response to hardware or software reporting a security violation depends on the state of the [Security state machine](#), and the security policy that software has configured in SNVS. Software configures the SNVS security policy by writing to the following registers:

- [SNVS_HP Security Interrupt Control Register \(HPSICR\)](#) - enables or disables generating the security interrupt when specific security violation events are detected
- [SNVS_HP Security Violation Control Register \(HPSVCR\)](#) - configures specific security violation event inputs as fatal or non-fatal or disabled
- [SNVS_HP High Assurance Counter IV Register \(PHACIVR\)](#) - sets the delay between the occurrence of an SSM transition from soft fail state to hard fail state

- [SNVS_HP Lock Register \(HPLR\)](#) - locks part or all of the security configuration set in SNVS_HP
- [SNVS_LP Control Register \(LPCR\)](#) - configures certain LP security event responses:
 - whether GPRs are zeroized in response to a security violation or tamper event
 - whether an LP power glitch alerts the PMIC
 - whether the RTC stops counting when a security violation occurs
 - whether a wake-up interrupt is requested when an LP section security event occurs
- [SNVS_HP Security Violation Control Register \(HPSVCR\)](#) - configures whether specific HP section security violation event inputs cause an LP section security violation, which zeroizes sensitive data in the LP section
- [SNVS_LP Tamper Detectors Configuration Register \(LPTDCR\)](#) - configures LP section analog and digital tamper detector sources
- [SNVS_LP Lock Register \(LPLR\)](#) - locks part or all of the security configuration set in SNVS_LP

If the SNVS SSM is in the Trusted or Secure state, the occurrence of a security violation triggers a transition to the Soft Fail state. The transition to Soft Fail state is reported to DCP, which will also take certain response actions.

The consequences of an SSM transition to Soft Fail include:

- Zeroization of the JDKEK, TDKEK and TDSK values in DCP
 - Zeroization of the JDKEK effectively zeroizes all normal Black Keys
 - Zeroization of the TDKEK effectively zeroizes all Trusted Black Keys
 - Zeroization of the TDSK invalidates all Trusted Descriptors
- Zeroization of the ZMK value in SNVS_LP
 - Zeroization of the ZMK effectively zeroizes all blobs encrypted using the ZMK or using ZMK XOR OTPMK
- Zeroization of the GPR value in SNVS_LP
 - Zeroization of the GPR protects the confidentiality of key or other secret data stored in the GP register
- Lock out of the master key (OTPMK, ZMK or OTPMK XOR ZMK) until the next POR (and successful secure boot)
 - Lock out prevents Trusted or Secure blobs from being decrypted until POR (and successful secure boot)

Note that software can continue to run following a transition to Soft Fail, and both SNVS and DCP remain operable. However, it is inadvisable to continue normal operation until the SoC has been reset. Whatever event triggered the transition from Secure or Trusted to Soft Fail state may have left the SoC in a vulnerable condition. Also, as explained below, after such a transition certain DCP security functions will be inoperable or unsecure.

- Attempting to decrypt Secure and Trusted blobs following a transition to Soft Fail will produce an error indication. (Since the Master Key is unavailable, the signature on the blobs will be incorrect.)
- Attempting to use pre-Fail-transition CCM-encrypted Black Keys will produce an error indication. (The JDKEK and TDKEK values will be incorrect now that they have been zeroized, consequently the CCM cryptographic signature will be incorrect).
- Attempting to use pre-Fail-transition Trusted Descriptors will produce an error indication. (The TDSK value will be incorrect now that it has been zeroized, consequently the cryptographic signature will be incorrect).
- Using pre-Fail-transition ECB-encrypted Black Keys will not produce an error indication because there is no cryptographic signature on ECB-encrypted Block Keys. (Using these keys will simply yield an incorrect result.)
- Encrypting blobs will not produce an error, but these blobs will not be effectively protected. (Since the Master Key is unavailable, the test key will be used.)
- Generating either CCM-encrypted or ECB-encrypted Black Keys will not produce an error, but these Black Keys will not be effectively protected. (Since the JDKEK and TDKEK have been zeroized, Black Keys will be encrypted with a known value.)

Consequently, operation following a transition from Secure or Trusted to Soft Fail state should be limited to recovery and debugging. In response to a Soft Fail, a security violation interrupt service routine should read the SNVS HP_Status Register and HP_Security Violation Status Register and write their values to NVRAM to log the occurrence and root cause of the security violation.

10.4.4 Tamper detects

This device includes OEMs have the ability to define system-level, physical security policies that cause violations of those physical security policies to be reported to SNVS via the SoC's tamper detection input signals. Examples of potential OEM-defined external tamper detection circuits include contact switches or light-sensitive switches that detect when the system's case has been opened, or circuitry to detect out-of range temperature, voltage or clock frequency. The OEM-defined external tamper detection circuitry needs to maintain the chip's tamper detect inputs at the voltage specified in the device's datasheet, because the SNVS will react to a voltage drop in the signal. Detection of an external tamper event is reported in the [SNVS_HP Security Violation Status Register \(HPSVSR\)](#) and [SNVS_LP Status Register \(LPSR\)](#).

The definition of a tamper event is up to the system developer. It is the responsibility of the system-level external tamper detection circuitry to maintain the tamper detection input pins at the proper voltage unless a tamper event occurs. Upon detection of a tamper

Security violation policy

event, the external logic should drive the tamper detect signals low, and it is recommended that a pull-down resistor be used to ensure the signals go low as quickly as possible if the inputs are cut. Since the tamper detect inputs are independent there is no requirement to implement all of them, however any tamper detect that isn't used must be tied so that it doesn't create false positives. If not used, TMP_DETECT_B should be connected to constant OVDD and LP_TMP_DETECT_B should be connected to a constant VDD. TMP_DETECT_B is sampled upon negation of PORESET_B, meaning that the external tamper detection logic must follow the OVDD sequencing requirements in the device hardware specification. Similarly, LP_TMP_DETECT_B is sampled upon negation of PORESET_B, meaning that the external low-power tamper detection logic must follow the VDD_LP sequencing requirements in the device hardware specification.

10.4.5 SNVS_LP security violation/tamper policy

The tables below provide information about the SNVS_LP security violation/tamper policy.

Table 10-4. SNVS_LP security violation sources

Security violation/tamper source	Default behavior ¹	Configuration options	Comments
Power glitch violation	Enable	-	Asserted when value in the Power Glitch Detector Register is different from the predefined hardwired value.

1. Default behavior refers to the setting after LP POR, before the LP Security Violation policy is configured

When an LP security violations is generated, the violation is reported to the SNVS_HP section. The source of the violation is recorded in the LP status register.

10.4.6 Power glitch detector (PGD)

SNVS_LP incorporates a PGD mechanism to detect an attack in which someone power gates the SNVS_LP power supply for short periods to cause the LP control, status, and secure counter values to change. This mechanism detects and alerts the system whenever power supply glitches endanger SNVS_LP register values. The mechanism works as follows:

1. The PGD register (LPPGDR) is loaded to the known specific value 4173_6166h as part of the SNVS initialization process.
2. This register's value is compared to a hardwired value to detect a glitch in the LP power supply.

Power glitch detection is always enabled and cannot be disabled. Immediately after LP POR, this register is cleared and a power glitch violation is asserted. Therefore, before programming any feature in the SNVS users should set a proper value into LPPGDR (see [SNVS_LP Power Glitch Detector Register \(LPPGDR\)](#)) and should clear the power glitch record in the LP status register (see [SNVS_LP Status Register \(LPSR\)](#)).

The following figure shows the PGD mechanism.

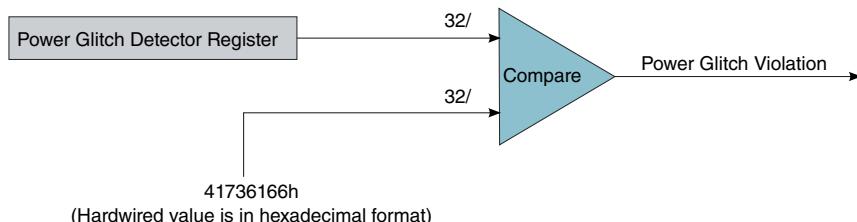


Figure 10-3. Power glitch detector

10.4.7 High Assurance Counter

If a security event occurs that transitions SNVS to the Soft Fail state, the software that is intended to respond to this transition may fail to act because it has become corrupted. SNVS implements a software "deadman" switch that is intended to deal with this case. The deadman switch consists of the High Assurance Counter (HAC) and the various other registers and register bit fields that control the HAC.

If enabled, the HAC begins counting down toward 0 when the SNVS Security State Machine (SSM) transitions to Soft Fail state. If software does not stop the count before it reaches 0, the security state machine will transition to the Hard Fail state. Transitioning to Hard Fail state is intended to cause the chip to reset.

If the HAC will be used, software should write the HAC initial value to the High Assurance Counter IV Register and then write a 1 to the HAC_LOAD (High Assurance Counter Load) bit and a 1 to the HAC_EN (High Assurance Counter Enable) in the HP Command Register. This will load the initial value into the HAC Register and enable the HAC for counting. Once the HAC register is initialized and enabled, software should write a 1 to the HAC_L (High Assurance Configuration Lock) bit in the HP Lock Register to prevent malicious or corrupted software from altering the HAC configuration.

If a security event causes the sSSM to transition to SOFT FAIL state the HAC will start counting, one count per system clock, from the initial value down to 0. While the SSM is in SOFT FAIL state software can halt the count by writing a 1 to the HAC_STOP bit in the HP Command Register. If software later writes a 0 to HAC_STOP the count will resume at the current value in the HAC. If software writes a 1 to the HAC_CLEAR bit in

the HP Command Register, the HAC will be cleared, regardless of the state of the SSM. If the SSM is in the Soft Fail state at that time the SSM will immediately transition to the Hard Fail state. If the SSM is in any other state when the HAC is cleared the SSM will transition to Hard Fail immediately after transitioning to Soft Fail (assuming the HAC is enabled).

10.5 Runtime Procedures

SNVS implements a number of features that are intended to be accessed by software at runtime (as opposed to accessed at boot time). These features include:

- Real Time Clock (see [SNVS_HP Real Time Counter](#))
- Secure Real Time Clock (see [SNVS_LP Secure Real Time Counter \(SRTC\)](#))
- General Purpose Register (see [Using the General-Purpose Register](#))
- Monotonic Counter (see [Using the Monotonic Counter \(MC\)](#))

Procedures for using these features are described in the following sections.

10.5.1 Using SNVS Timer Facilities

SNVS incorporates timer facilities that can optionally generate an interrupt at a specified time. As described in the following sections, SNVS_HP incorporates a Real Time Counter that is available for general use, and SNVS_LP incorporates a Secure Real Time Counter intended for security applications.

10.5.1.1 SNVS_HP Real Time Counter

SNVS_HP implements a real time counter that can be read or written by any application; it has no privileged software access restrictions. When the chip is powered down the RTC is not active and it is reset at chip POR. The RTC can be used to generate a functional interrupt request either at a specific time, or at a specific frequency, or both. To generate an interrupt request at a specific time HPTA_EN is set to 0, the desired time is written to HPTA_MS and HPTA_LS and then HPTA_EN is set to 1. HPTA_EN, HPTA_MS and HPTA_LS can be written by any software that has access to SNVS registers; there are no privileged access restrictions. The counter can be synchronized to the SNVS_LP SRTC by writing to the HP_TS bit of SNVS_HP Control Register. This is particularly useful if the SNVS_LP is powered from an uninterrupted power source (e.g. a coin cell battery) because the RTC can then be set from a chip-internal time source.

10.5.1.2 SNVS_LP Secure Real Time Counter (SRTC)

SNVS_LP implements a secure real time counter. The SRTC differs from the Realtime Counter (RTC) implemented in the HP section in the following ways:

- If the SNVS_LP power input is connected to an uninterrupted power supply (e.g. coin cell battery) the SRTC retains its state and continues counting even when the main chip is powered down.
- When the RTC reaches its maximum value it simply rolls over to all 0s. The SRTC is a non-rollover counter, which means that the SRTC does not rollover to all zeros when it reaches the maximum value of all ones. Instead, a time rollover indication is generated to the SNVS_LP tamper monitor, which generates a security violation and interrupt.
- The RTC can be written at any time. The SRTC can be locked so that a new value cannot be written into the SRTC.
- The RTC can be synchronized to the SRTC by writing to the HP_TS bit of SNVS_HP Control Register.
- The SRTC can be calibrated so that its count frequency tracks some other time source.
- The SRTC can be marked as valid or invalid, so software knows whether the SRTC value can be trusted.
- The SRTC can be configured to capture the time of a security violation.

The SRTC is intended for security applications that require real time. Some examples are:

- data rights management schemes in which the rights expire at a certain date and time
- audit logs that record the date and time of security-relevant events
- security protocols that rely on real time to ensure freshness or for event ordering

To ensure that the SRTC cannot be modified in order to circumvent time-based security policies, the SRTC can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set. If the SRTC will be used for security purposes, at SNVS_LP POR trusted software (e.g. boot firmware) should check if SRTC is enabled and valid (SRTC_ENV=1). If SRTC_ENV=0 the trusted software should set the SRTC from a trustworthy time source and then set SRTC_ENV, SRTC_SL and SRTC_HL to 1. When SRTC_SL=1 the SRTC cannot be written until main chip reset. When SRTC_HL=1 the SRTC cannot be written until SNVS_LP POR.

The SRTC can be used to generate a functional interrupt request at a particular time by setting the desired time in the LP Time Alarm register (LPTAR) and setting LPTA_EN to 1. Note that the functional interrupt request is shared by both the LP time alarm and the

HP time alarm, so the interrupt handler for the SNVS functional alarm should check the HPTA bit in the HPSR and/or the LPTA in the LPSR to distinguish between these two alarm sources.

The SRTC counting speed can be adjusted to match the speed of some other time source via a calibration mechanism. When the LPCALB_EN, LPCALB_SL, and LPCALB_HL bits are 0 a value can be written into the LPCALB_VAL field of the LP Control Register. When LPCALB_EN is 1, every 32768 ticks of the SRTC the adjustment value specified in the LPCALB_VAL field will be added to the SRTC. If needed, at LP POR the LPCALB_VAL field should be set by trusted software and then LPCALB_EN set to 1. Whether the SRTC calibration mechanism is used or not, trusted software should set LPCALB_SL, and LPCALB_HL bits to 1 to prevent misuse of the calibration mechanism.

10.5.1.3 RTC/SRTC control bits setting

All SNVS registers are programmed from the register bus, consequently any software-initiated changes are synchronized with the IP clock. Several registers can also change synchronously with the RTC/SRTC clock after they are programmed. To avoid IP clock and RTC/SRTC clock synchronization issues, the following values can be changed only when the corresponding function is disabled.

Table 10-5. RTC/SRTC synchronized values list

Function	Value/register	Control bit setting
HP section		
HP Real Time Counter	HPRTCMR and HPRTCLR Registers	RTC_EN = 0 : HPRTCMR/HPRTCLR can be programmed RTC_EN = 1 : HPRTCMR/HPRTCLR cannot be programmed
HP Time Alarm	HPTAMR and HPTALR Registers	HPTA_EN = 0 : HPTAMR/HPTALR can be programmed HPTA_EN = 1 : HPTAMR/HPTALR cannot be programmed
LP section		
LP Secure Real Time Counter	LPRTCMR and LPRTCLR Registers	SRTC_ENV = 0 : LPRTCMR/LPRTCLR can be programmed SRTC_ENV = 1 : LPRTCMR/LPRTCLR cannot be programmed
LP Time Alarm	LPTAR Register	LPTA_EN = 0 : LPTAR can be programmed LPTA_EN = 1 : LPTAR cannot be programmed

Use the following steps to program synchronized values:

1. Check the enable bit value. If set, clear it.
2. Verify that the enable bit is cleared. There are two reasons to verify the enable bit's setting:
 - Enable bit clearing does not happen immediately; it takes three IP clock cycles and two RTC/SRTC clock cycles to change the enable bit's value.
 - If the enable bit is locked for programming, it cannot be cleared.
3. Program the desired value.
4. Set the enable bit; it takes three IP clock cycles and two RTC/SRTC clock cycles for the bit to set.

NOTE

Incrementing the value programmed into RTC/SRTC registers by two compensates for the two RTC/SRTC clock cycle delay that is required to enable the counter.

10.5.1.4 Reading RTC and SRTC values

Software should follow the following procedure to ensure that it has read correct data from the RTC (HPRTCMR and HPRTCLR) and SRTC (LPSRTCMR and LPSRTCLR) registers:

- Read the most-significant half and the least-significant half of the RTC/SRTC and then read both halves again. If the values read are the same both times, the value is correct.
- If the two consecutive pairs of reads yield different results, perform two more reads.

The worst case scenario may require three sessions of two consecutive pairs of reads.

There are several reasons that the values may be incorrectly read initially:

- Synchronization issues between the RTC/SRTC clock and the system clock
- Since the counter continues to increment, there may be a carry from the least-significant 32-bits to the most-significant bits in between reading the two halves of the counter

10.5.2 Using Other SNVS Registers

The sections below describe how to use the General Purpose Register.Monotonic Counter.The sections below describe how to use the General Purpose Register and the Monotonic Counter.

10.5.2.1 Using the General-Purpose Register

SNVS implements a 128-bit general-purpose register allows software to store a small amount of data. To maintain backward compatibility with versions of SNVS that implement only a 32-bit general purpose register, the most-significant word of the general purpose register is aliased to the legacy address. The data in the GPR will be retained during system power-down mode as long as the SNVS_LP remains powered by an uninterrupted power source (e.g. coin cell battery). If LP Control Register GPR_Z_DIS bit is 0, the GPR will be zeroized if a tamper event occurs.

10.5.2.2 Using the Monotonic Counter (MC)

The following figure shows the MC and its rollover security violation.

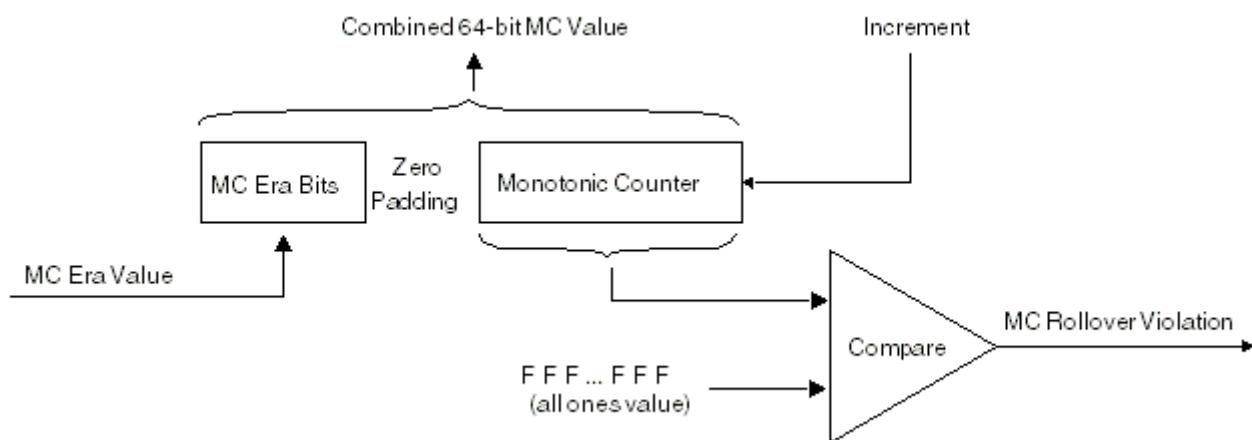


Figure 10-4. SNVS_LP monotonic counter

Some security applications require a monotonic counter (MC) that cannot be returned to any previous value during the product's lifetime. For instance, the MC can be used to detect replay of a cryptographic blob by including the current monotonic counter value in the blob when it is encrypted, and checking this value against the current value in the MC register when the blob is decrypted. If more than one blob must be replay-protected, when any blob is to be updated the MC value would be incremented and all the blobs would be refreshed with the updated MC value. Alternatively, software monotonic

counters could be maintained in a "monotonic counter blob", which would be replay-protected via the hardware MC. In this case any time that a blob is to be updated its software monotonic counter would be incremented, the MC would be incremented, and the monotonic counter blob would be refreshed with the updated value of the software monotonic counter and the updated MC value. If the application can tolerate a limited window for replay, the overhead could be reduced by only updating the monotonic counter blob once an hour, or perhaps once a day. Software could create even more elaborate anti-replay mechanisms using a tree of monotonic counter blobs, with the hardware MC at the root.

Because the MC cannot be allowed to repeat a value, it cannot set to a lower value than it currently holds. In the unlikely case that it reaches its maximum value, it does not rollover. Instead, a monotonic counter rollover indication is generated to the SNVS_LP tamper monitor. This generates a security violation to the SSM and an interrupt to the host processor.

The MON_COUNTER fields of the MC register are implemented in flip flops within the LP section. If LP power is lost, the MON_COUNTER value will be lost. To preserve monotonicity in this event, the most significant bits of MC (the MC_ERA_BITS field) are derived from fuses. This ensures that the MC_ERA_BITS value is preserved across LP section power failures. The next time that the chip powers up following an LP section power failure, the chip's boot firmware will note that the monotonic counter value is invalid and will blow another of the fuses that drive the MC_ERA_bits field. This will result in a larger value in the MC_ERA MC field and since this field forms the most-significant bits of MC, this guarantees that the new value of MC is greater than any of its past values.

10.6 Configuring Master Key checking and control

During cryptographic blob operations DCP uses a key derivation function to create a 256-bit blob key encryption key (BKEK). The BKEK is used to encrypt or decrypt a 256-bit random key that is actually used to encrypt or decrypt the content of the blob. Note that the BKEK is nonvolatile so that blobs generated during one power-on cycle can be decrypted during later power-on cycles. One of the inputs to the BKEK key derivation function is the Master Key. When DCP is in the non-Secure or Fail mode, a test value is used for this Master Key. But when DCP is in the Trusted or Secure mode the key value used is the Master Key supplied by the SNVS. Software configures SNVS via the LP Master Key Control Register's MASTER_KEY_SEL (Master Key Select) field so that the Master Key supplied to DCP comes from one of the following sources:

- One Time Programmable Master Key (OTPMK): This value is permanently burned into fuses. The 256-bit value includes an embedded 9-bit Hamming code so that

hardware can verify the integrity of the key. This code is capable of detecting all single, double, and triple bit errors in the key value. In addition the hardware checks for an all-zero value of the OTPMK. If the OTPMK Hamming code check fails or the OTPMK is not programmed at all (all 0s), the hardware will report a "bad key" error, which will prevent the chip from reaching the Trusted/Secure state. The construction of the Hamming code is described in [Error Code for the OTPMK](#).

- Zeroizable Master Key (ZMK): This value is stored in the ZMK registers of the low power (LP) section of the SNVS. The ZMK value is retained when the main SoC power is off because power is supplied to the SNVS LP section from a coin cell battery. The ZMK value is automatically erased in response to fatal security violations. OEMs have two options for provisioning a ZMK:
 - Software Programming: Software can write directly to the SNVS's ZMK registers. Once software has finished writing all of the ZMK registers, software sets the ZMK_Valid bit. The value to be written into the ZMK can be determined by any appropriate method. For instance, a random value could be obtained from the RNG, or software could program the ZMK registers with a secret value shared among the OEM's devices. Unlike the OTPMK, the ZMK value does not include an embedded Hamming code. Instead, additional ECC protection bits are automatically generated by hardware and stored in the SNVS LP section.
 - Hardware Programming: SNVS and DCP can be configured so that a 256-bit random value is generated in the random number generator and loaded directly into the SNVS's ZMK registers via a private hardware interface. Additional ECC protection bits are automatically generated by the SNVS and stored in the SNVS LP section along with the ZMK.
- Combined Master Key (CMK): SNVS can be configured so that the Master Key is the bitwise XOR of the OTPMK registers and the ZMK registers. This bitwise XOR is called the CMK.

When SNVS detects a fatal security violation, SNVS signals the DCP to switch to the Test key instead of the Master Key value when encrypting or decrypting blobs. Until the next successful secure boot DCP will not be able to decrypt any blobs that were generated while DCP was in Trusted or Secure mode. Note that the Master Key is used by the DCP only as a key derivation key, and only during blob operations. It is never used to directly encrypt or decrypt user selected data. Within the DCP the Master Key value is combined with a number of other inputs to derive a blob key encryption key (BKEK). During blob creation the BKEK is used to encrypt a 256-bit blob key obtained from the RNG. This random blob key is used to encrypt the content of the blob. The encrypted blob key is stored as part of the blob. During blob decryption the BKEK is recreated so that the encrypted blob key can be decrypted, and then used to decrypt the blob content.

10.6.1 Error Code for the OTPMK

The Hamming code used for the 256-bit OTPMK has nine code bits. It can detect all 1, 2, and 3 bit errors in the codeword. It also detects most, but not all, errors of more than 3 bits. Therefore, the OTPMK actually contains 247 random bits.

Numbering the bits in the OTPMK from zero to 255, the code bits are bits 0, 1, 2, 4, 8, 16, 32, 64, and 128. All remaining bits are data bits. Each of these code bits is the XOR of a subset of the bits in the entire code word.

To determine which bits are used to form each code bit, look at the binary representation of the bit position of each code bit (ignoring bit zero for the time being) in the following table.

Table 10-6. Error codes

Code bit	Binary representation
0	00000000
1	00000001
2	00000010
4	00000100
8	00001000
16	00010000
32	00100000
64	01000000
128	10000000

For code bit number 1, there is a single one in its binary representation. This code bit is the XOR of all bit positions that also have a one in this same point in their binary representation (i.e., all odd bit positions). The same is true for the other code bits, excepting code bit 0. For example, code bit 2 is the XOR of bits 3, 6, 7, 10, 11, 14, 15,..., 254, 255, and code bit 128 is the XOR of all bits from 129 through 255 inclusive. After all of the other code bits have been calculated, code bit zero is simply the XOR of all of the other bits, including the code bits.

Another way of looking at this is to ask which code bits does a data bit affect. If we look at the binary representation of the bit position of a data bit, the 1s represent the code bits that this data bit affects. For example, data bit 99 (01100011) is XORed into code bits 1, 2, 32, 64, and bit 0 (the overall parity bit).

10.6.2 Provisioning the Zeroizable Master Key

The ZMK is programmed after or during initialization of the SNVS_LP. The ZMK can be provisioned either by software or by hardware. After the ZMK has been provisioned, the value is retained even when power is off in the most of the chip. The SNVS's LP section (SNVS_LP) includes the following registers for storing and locking the Zeroizable Master Key (ZMK).

Register	Overview
SNVS_LP Zeroizable Master Key Registers	Eight 32-bit registers store a 256-bit value called the Zeroizable Master Key (ZMK). The ZMK provides essentially the same function as the OTPMK from the Security Fuse Processor, except that the ZMK value can be zeroized if a security violation is detected. Upon detection of a security violation both the ZMK and the OTPMK are rendered unavailable during the current boot cycle, but in addition the ZMK is zeroized, which renders the ZMK value unavailable upon all subsequent boot cycles.
SNVS_LP Lock Register	This register controls read and write access to all other SNVS_LP registers. Once a bit in the SNVS_LP Lock Register is set, it cannot be cleared without a device reset.
SNVS_LP Master Key Control Register	SNVS_LP Master Key Control determines whether the ZMK, the OTPMK, or the XOR of the two, will be used by the DCP when creating cryptographic blobs. This register also: <ul style="list-style-type: none"> • controls the programming mode for the ZMK (software, or hardware via Random Number Generator) • stores the ZMK's Error Correcting Code • implements a valid bit that indicates that the ZMK has been provisioned
SNVS_LP Status Register	Provides status information about the SNVS Low Power section's security state, including hardware security violations that are directly detected within the SNVS_LP. The ZMK cannot be provisioned if the SNVS_LP is in a Fail state.

The SNVS_LP Master Key Control Register gives OEMs the option of directly writing a ZMK value to the ZMK Registers, or commanding the SNVS hardware to load a ZMK by acquiring a 256-bit random number from the RNG. When the SNVS_LP loads its ZMK via the RNG, hardware also calculates the associated Hamming code and loads it into the ZMK_ECC_Value field of the SNVS_LP Master Key Control Register.

To provision ZMK by software do the following:

- Verify that SNVS_LPMKCR [ZMK_HWP] is not set
- Verify that the ZMK registers are not locked for reads and writes. Verify that SNVS_HPLR [ZMK_WSL], [ZMK_RSL] or SNVS_LPLR [ZMK_WHL], [ZMK_WHL] are not set.
- Write key value to the ZMK registers
- Verify that the intended key value is written
 - Set SNVS_LPMKCR [ZMK_VAL] to indicate the ZMK is ready to be used the DCP.
 - (Optional but recommended) Set SNVS_LPMKCR [ZMK_ECC_EN] to enable ZMK error correction code verification. Software can verify that the correct nine-bit codeword is generated by reading ZMK_ECC_VALUE field

- (Optional but recommended) Block software read accesses to the ZMK registers and ZMK_ECC_VALUE field by setting SNVS_HPLR [ZMK_RSL] and SNVS_LPLR [ZMK_RHL]
- (Optional but recommended) Block software write accesses to the ZMK registers by setting SNVS_HPLR [ZMK_WSL] and SNVS_LPLR [ZMK_WHL]
- Set SNVS_HPCOMR [MKS_EN] and SNVS_LPMKCR [MASTER_KEY_SEL] to select combination of OTPMK and ZMK to be provided to the DCP
- (Optional but recommended) Block software write accesses to the MASTER_KEY_SEL field by setting MKS lock bit
- (Optional but recommended) Block software write accesses to SNVS_LPMKCR [MASTER_KEY_SEL] by setting SNVS_HPLR [MKS_SL].

To provision ZMK by hardware do the following:

- Verify that the ZMK registers are not locked for writes. Check that SNVS_HPLR [ZMK_WSL], or SNVS_LPLR [ZMK_WHL] are not set
- Set SNVS_LPMKCR [ZMK_HWP]
- Set SNVS_HPCOMR [PROG_ZMK]
- Poll for SNVS_LPMKCR [ZMK_VAL] bit to be set. This bit is set by hardware at the end of the ZMK programming cycle
- (Optional but recommended) Set SNVS_LPMKCR [ZMK_ECC_EN] to enable ZMK error correction code verification by hardware. Note that the ZMK Registers and ZMK_ECC_VALUE field cannot be read by software in the hardware programming mode, meaning hardware ZMK_ECC checking is the only way to determine if the ZMK is corrupted.
- (Optional but recommended) Block hardware programming option of the ZMK Registers by setting SNVS_LPLR [ZMK_WHL]
- Set SNVS_HPCOMR [MKS_EN] and SNVS_LPMKCR [MASTER_KEY_SEL] to select combination of OTPMK and ZMK to be provided to the DCP
- (Optional but recommended) Block software write accesses to SNVS_LPMKCR [MASTER_KEY_SEL] by setting SNVS_HPLR [MKS_SL].

If the ZMK is provisioned by hardware, the ZMK Registers and the ZMK_ECC_VALUE field in the SNVS_LP Master Key Control Register cannot be read by software.

10.7 Reset and Initialization of SNVS

SNVS is implemented in two sections (HP and LP) that both must be initialized by software following POR. If the SNVS_LP is powered by an uninterrupted power source that is separate from main SoC power, then SNVS can operate in either of two modes, depending upon whether the main SoC power is on or off. During main SoC power-down

SNVS_HP is powered-down, but SNVS_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode SNVS_LP keeps its registers' values and monitors the SNVS_LP tamper detection inputs, but the LP registers cannot be read or written. During main SoC power-up the isolation of SNVS_LP is disabled and both SNVS_HP and SNVS_LP are powered from the main SoC power. Both LP and HP registers can be read and written (locks and privilege modes permitting). Signals between the SNVS_HP and SNVS_LP sections are enabled and all SNVS functions are operational.

Since the HP and LP sections reside in different power domains, the POR for the two sections can occur at different times. If the SNVS_LP section remains powered by an uninterrupted power source when the main SoC power is off, SNVS_LP is initialized rarely, typically once when the device is first powered on and again whenever the battery is replaced. During main SoC power-up the isolation of SNVS_LP is disabled and both SNVS_HP and SNVS_LP are powered from the main SoC power. Signals between the SNVS_HP and SNVS_LP sections are enabled and all SNVS functions are operational. The SNVS_HP section is powered from the main SoC power, so it must be initialized whenever the device is powered on. If the SNVS_LP section is powered from the main SoC power rather than from an uninterrupted power source, the SNVS_LP section must also be initialized at SoC POR.

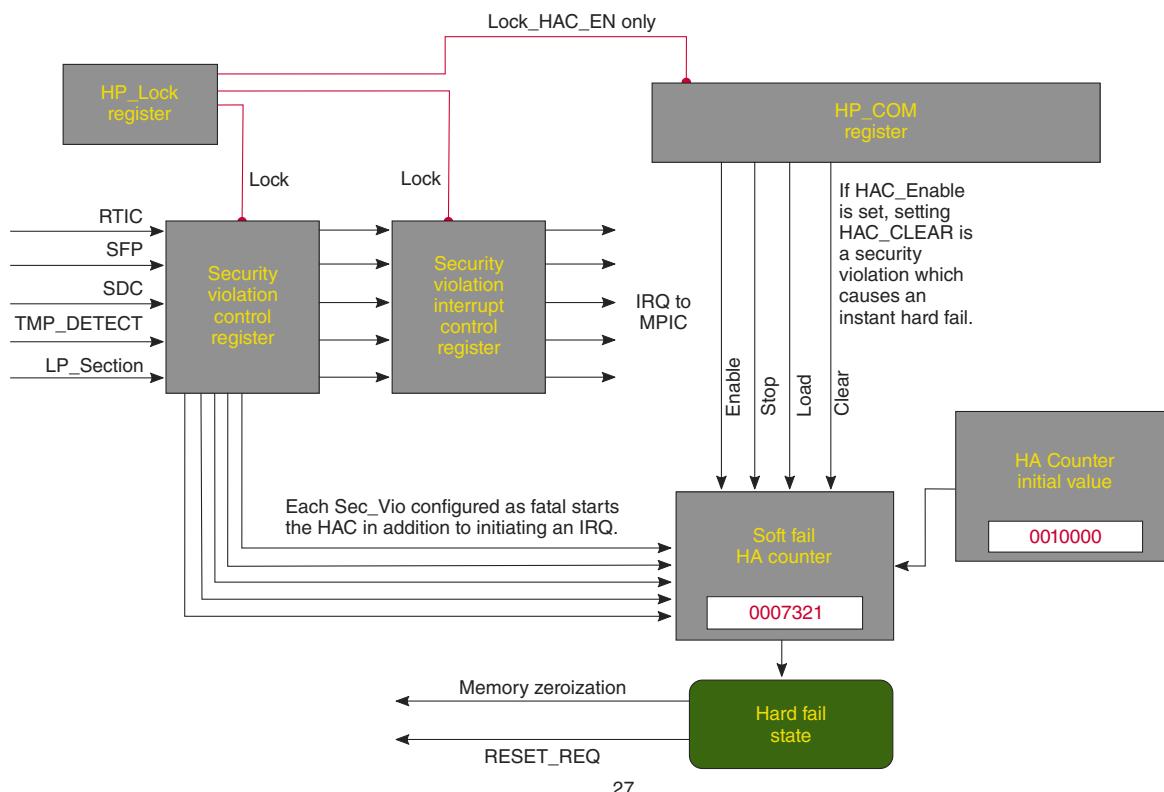
Initializing the LP section The following steps should be completed to properly initialize the SNVS LP section (required only on LP POR, i.e. when the battery is replaced):

1. Program the Power Glitch Detector and clear the power glitch record in the LP status register
2. If the SRTC will be used, set the Secure Real Time Clock
3. If the ZMK will be used, provision the ZMK
4. If the Monotonic Counter will be used, burn an additional Monotonic Counter Era bit in the fuse bank and reset the Monotonic Counter

Initializing the HP section The following steps should be completed to properly initialize the SNVS HP section (following successful completion of secure boot):

1. Perform normal or secure boot to put the SNVS into a functional state (Non-secure, Trusted, Secure)
2. At SNVS_LP POR a power glitch violation will be asserted. Therefore at the first HP POR following an LP POR software should write the proper initialization value (41736166h) into the LPPGDR and clear the power glitch record in the LP status register. See [Power glitch detector \(PGD\)](#) for more details.
3. Transition SSM from trusted state to secure state (if not already done by HAB).
4. Enable security violations and interrupts in HPSVCR and HPSICR registers
5. Program SNVS general functions/configurations
6. Select Master Key (OTPMK, ZMK or XOR of the two. Default is OTPMK.)

7. Set lock bits. The ms bit of the SNVS_HP lock register should be set before starting any functional operation. Setting this bit prevents further changes to the Master Key selection.



27

Figure 10-5. Relationship Between the Registers

10.7.1 Power glitch detector (PGD)

SNVS_LP incorporates a PGD mechanism to detect an attack in which someone power gates the SNVS_LP power supply for short periods to cause the LP control, status, and secure counter values to change. This mechanism detects and alerts the system whenever power supply glitches endanger SNVS_LP register values. The mechanism works as follows:

1. The PGD register (LPPGDR) is loaded to the known specific value 4173_6166h as part of the SNVS initialization process.
2. This register's value is compared to a hardwired value to detect a glitch in the LP power supply.

Power glitch detection is always enabled and cannot be disabled. Immediately after LP POR, this register is cleared and a power glitch violation is asserted. Therefore, before programming any feature in the SNVS users should set a proper value into LPPGDR (see [SNVS_LP Power Glitch Detector Register \(LPPGDR\)](#)) and should clear the power glitch record in the LP status register (see [SNVS_LP Status Register \(LPSR\)](#)).

The following figure shows the PGD mechanism.

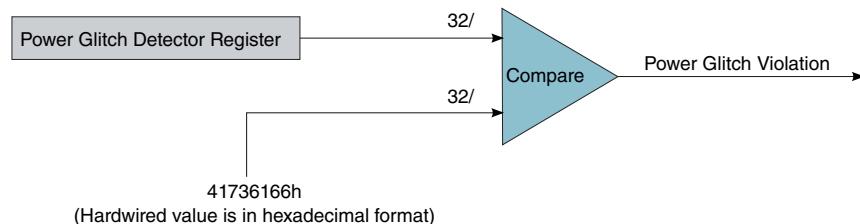


Figure 10-6. Power glitch detector

10.7.2 SNVS_LP Secure Real Time Counter (SRTC)

SNVS_LP implements a secure real time counter. The SRTC differs from the Realtime Counter (RTC) implemented in the HP section in the following ways:

- If the SNVS_LP power input is connected to an uninterrupted power supply (e.g. coin cell battery) the SRTC retains its state and continues counting even when the main chip is powered down.
- When the RTC reaches its maximum value it simply rolls over to all 0s. The SRTC is a non-rollover counter, which means that the SRTC does not rollover to all zeros when it reaches the maximum value of all ones. Instead, a time rollover indication is generated to the SNVS_LP tamper monitor, which generates a security violation and interrupt.
- The RTC can be written at any time. The SRTC can be locked so that a new value cannot be written into the SRTC.
- The RTC can be synchronized to the SRTC by writing to the HP_TS bit of SNVS_HP Control Register.
- The SRTC can be calibrated so that its count frequency tracks some other time source.
- The SRTC can be marked as valid or invalid, so software knows whether the SRTC value can be trusted.
- The SRTC can be configured to capture the time of a security violation.

The SRTC is intended for security applications that require real time. Some examples are:

- data rights management schemes in which the rights expire at a certain date and time

- audit logs that record the date and time of security-relevant events
- security protocols that rely on real time to ensure freshness or for event ordering

To ensure that the SRTC cannot be modified in order to circumvent time-based security policies, the SRTC can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set. If the SRTC will be used for security purposes, at SNVS_LP POR trusted software (e.g. boot firmware) should check if SRTC is enabled and valid (SRTC_ENV=1). If SRTC_ENV=0 the trusted software should set the SRTC from a trustworthy time source and then set SRTC_ENV, SRTC_SL and SRTC_HL to 1. When SRTC_SL=1 the SRTC cannot be written until main chip reset. When SRTC_HL=1 the SRTC cannot be written until SNVS_LP POR.

The SRTC can be used to generate a functional interrupt request at a particular time by setting the desired time in the LP Time Alarm register (LPTAR) and setting LPTA_EN to 1. Note that the functional interrupt request is shared by both the LP time alarm and the HP time alarm, so the interrupt handler for the SNVS functional alarm should check the HPTA bit in the HPSR and/or the LPTA in the LPSR to distinguish between these two alarm sources.

The SRTC counting speed can be adjusted to match the speed of some other time source via a calibration mechanism. When the LPCALB_EN, LPCALB_SL, and LPCALB_HL bits are 0 a value can be written into the LPCALB_VAL field of the LP Control Register. When LPCALB_EN is 1, every 32768 ticks of the SRTC the adjustment value specified in the LPCALB_VAL field will be added to the SRTC. If needed, at LP POR the LPCALB_VAL field should be set by trusted software and then LPCALB_EN set to 1. Whether the SRTC calibration mechanism is used or not, trusted software should set LPCALB_SL, and LPCALB_HL bits to 1 to prevent misuse of the calibration mechanism.

10.7.3 Initialization Checklists

SNVS is implemented in two sections (HP and LP) that both must be initialized by software following POR. If the SNVS_LP is powered by an uninterrupted power source that is separate from main SoC power, then SNVS can operate in either of two modes, depending upon whether the main SoC power is on or off. During main SoC power-down SNVS_HP is powered-down, but SNVS_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode SNVS_LP keeps its registers' values and monitors the SNVS_LP tamper detection inputs, but the LP registers cannot be read or written. During main SoC power-up the isolation of SNVS_LP is disabled and both SNVS_HP and SNVS_LP are powered from the main SoC power.

Reset and Initialization of SNVS

Both LP and HP registers can be read and written (locks and privilege modes permitting). Signals between the SNVS_HP and SNVS_LP sections are enabled and all SNVS functions are operational.

Since the HP and LP sections reside in different power domains, the POR for the two sections can occur at different times. If the SNVS_LP section remains powered by an uninterrupted power source when the main SoC power is off, SNVS_LP is initialized rarely, typically once when the device is first powered on and again whenever the battery is replaced. During main SoC power-up the isolation of SNVS_LP is disabled and both SNVS_HP and SNVS_LP are powered from the main SoC power. Signals between the SNVS_HP and SNVS_LP sections are enabled and all SNVS functions are operational. The SNVS_HP section is powered from the main SoC power, so it must be initialized whenever the device is powered on. If the SNVS_LP section is powered from the main SoC power rather than from an uninterrupted power source, the SNVS_LP section must also be initialized at SoC POR.

Checklists for initializing the SNVS_HP and SNVS_LP sections appear in the following sections.

10.7.3.1 Checklist for Initialization of the SNVS HP

At SNVS HP POR, software must perform initialization actions in the SNVS HP section as indicated in the table below.

Initialization action (Some of these actions are performed in the boot firmware or software.)	Comments	See section
Configure fatal/nonfatal security violations	At HP POR all security violations default to disabled or nonfatal.	Configuring SNVS's response to a security violation or tamper event SNVS_HP Security Violation Control Register (HPSVCR)
Configure the High Assurance Counter	Can be skipped if HAC is not used. Defaults to HAC disabled at HP POR.	High Assurance Counter SNVS_HP High Assurance Counter IV Register (HPHACIVR) SNVS_HP High Assurance Counter Register (PHHACR) HAC_STOP, HAC_CLEAR, HAC_LOAD, and HAC_EN in SNVS_HP Command Register (HPCO MR) HAC_L in SNVS_HP Lock Register (HPLR)
Configure the SNVS interrupt policy	By default security violation interrupts are disabled at HP POR.	SNVS interrupts, alarms, and security violations

Table continues on the next page...

Initialization action (Some of these actions are performed in the boot firmware or software.)	Comments	See section
		SNVS_HP Security Interrupt Control Register (HPSICR)
Advance the Security State Machine	If software detects a security violation, force a transition to Soft Fail state, else cause a transition to Trusted state (and then to Secure state, if desired).	Security state machine SW_LPSV, SW_FSV, SW_SV, SSM_SFNS_DIS, SSM_ST_DIS and SSM_ST in SNVS_HP Command Register (HPCOMR)
Select Master Key input to CAAM	If ZMK is not used, select OTPMK (default), else select ZMK or CMK.	Configuring Master Key checking and control MASTER_KEY_SEL in SNVS_LP Master Key Control Register (LPMKCR)

10.7.3.2 Checklist for Initialization of the SNVS LP

At SNVS LP POR, software must perform initialization actions in the SNVS LP section as indicated in the table below. If the SNVS LP section is powered from the same power source as SNVS HP section, LP POR will occur at the same time as HP POR.

Initialization action (Some of these actions are performed in the boot firmware or software.)	Comments	See section
Configure tamper detection inputs	Can be skipped if tamper detection inputs are unused.	Configuring SNVS's response to a security violation or tamper event SNVS_HP Security Violation Control Register (HPSVCR) SNVS_LP Tamper Detectors Configuration Register (LPTDCR)
Burn additional MC_ERA fuse and initialize Monotonic Counter	Can be skipped if Monotonic Counter is unused. Should be skipped if SNVS LP is unpowered when main chip power is off.	Using the Monotonic Counter (MC) SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR) SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR) MC_SL in SNVS_HP Lock Register (HPLR) MC_HL in SNVS_LP Lock Register (LPLR) MC_ENV in SNVS_LP Control Register (LPCR)

Table continues on the next page...

SNVS register descriptions

Initialization action (Some of these actions are performed in the boot firmware or software.)	Comments	See section
Initialize Secure Real Time Counter	Can be skipped if Secure Real Time Counter is unused. Should be skipped if SNVS LP is unpowered when main chip power is off.	SNVS_LP Secure Real Time Counter (SRTC) SNVS_LP Secure Real Time Counter MSB Register (LPSRTCMR) SNVS_LP Secure Real Time Counter LSB Register (LPSRTCLR) SNVS_LP Time Alarm Register (LPTAR) SNVS_LP Time Alarm Register (LPTAR) LPCALB_VAL, LPCALB_EN, SRTC_INV_EN, LPTA_EN, SRTC_ENV in SNVS_LP Control Register (LPCR)
Program the Zeroizable Master Key	Can be skipped if Zeroizable Master Key is unused. Should be skipped if SNVS LP is unpowered when main chip power is off.	Provisioning the Zeroizable Master Key SNVS_LP Zeroizable Master Key Register (LPZMKR0 - LPZMKR7) SNVS_LP Master Key Control Register (LPMKCR) ZMK_RSL and ZMK_WSL in SNVS_HP Lock Register (HPLR) PROG_ZMK in SNVS_HP Command Register (HPCOMR) ZMK_ZERO in SNVS_HP Status Register (HPSR) ZMK_ECC_FAIL and ZMK_SYNDROME in SNVS_HP Security Violation Status Register (HPSVSR)

10.8 SNVS register descriptions

This section contains detailed register descriptions for the SNVS registers. Each description includes a standard register diagram and register table. The register table provides detailed descriptions of the register bit and field functions, in bit order.

SNVS registers consist of two types:

- Privileged read/write accessible
- Non-privileged read/write accessible

Privileged read/write accessible registers can only be accessed for read/write by privileged software. Unauthorized write accesses are ignored, and unauthorized read accesses return zero. Non-privileged software can access privileged access registers when the non-privileged software access enable bit is set in the SNVS_HP Command Register.

In addition, all privileged access registers (except HPSVSR and LPSR) can be written to only if the system security monitor is in one of the three functional states:

- Non-Secure
- Trusted
- Secure

Certain fields in the SNVS_HP Command Register can be written in non-functional system security monitor states (init, check, soft fail, and hard fail) as follows:

- SW_LPSV, SSM_ST, SW_SV, and SW_FSV can be written in check or soft fail state.
- The HAC_STOP bit can only be set in soft fail state but can be cleared in either soft fail or a functional state.
- HAC_LOAD, HAC_CLEAR bits and HPSVSR, LPSR Registers can be accessed in soft fail or a functional state.

The system security monitor state does not restrict read access to SNVS registers.

Non-privileged read/write accessible registers are read/write accessible by any software.

The LP register values are set only on LP POR and are unaffected by System (HP) POR. The HP registers are set only on System POR and are unaffected by LP POR.

The following table shows the SNVS main memory map.

10.8.1 SNVS Memory map

SNVS base address: 400D_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SNVS_HP Lock Register (HPLR)	32	RW	0000_0000h
4h	SNVS_HP Command Register (HPCOMR)	32	RW	0000_0000h

Table continues on the next page...

SNVS register descriptions

Offset	Register	Width (In bits)	Access	Reset value
8h	SNVS_HP Control Register (HPCR)	32	RW	0000_0000h
Ch	SNVS_HP Security Interrupt Control Register (HPSICR)	32	RW	0000_0000h
10h	SNVS_HP Security Violation Control Register (HPSVCR)	32	RW	0000_0000h
14h	SNVS_HP Status Register (HPSR)	32	W1C	8000_3000h
18h	SNVS_HP Security Violation Status Register (HPSVSR)	32	W1C	8000_0000h
1Ch	SNVS_HP High Assurance Counter IV Register (PHACIVR)	32	RW	0000_0000h
20h	SNVS_HP High Assurance Counter Register (PHACR)	32	RO	0000_0000h
24h	SNVS_HP Real Time Counter MSB Register (HPRTCMR)	32	RW	0000_0000h
28h	SNVS_HP Real Time Counter LSB Register (HPRTCLR)	32	RW	0000_0000h
2Ch	SNVS_HP Time Alarm MSB Register (HPTAMR)	32	RW	0000_0000h
30h	SNVS_HP Time Alarm LSB Register (HPTALR)	32	RW	0000_0000h
34h	SNVS_LP Lock Register (LPLR)	32	RW	0000_0000h
38h	SNVS_LP Control Register (LPCR)	32	RW	0000_0020h
3Ch	SNVS_LP Master Key Control Register (LPMKCR)	32	RW	0000_0000h
40h	SNVS_LP Security Violation Control Register (LPSVCR)	32	RW	0000_0000h
48h	SNVS_LP Tamper Detectors Configuration Register (LPTDCR)	32	RW	0000_0000h
4Ch	SNVS_LP Status Register (LPSR)	32	W1C	0000_0008h
50h	SNVS_LP Secure Real Time Counter MSB Register (LPSRTCMR)	32	RW	0000_0000h
54h	SNVS_LP Secure Real Time Counter LSB Register (LPSRTCLR)	32	RW	0000_0000h
58h	SNVS_LP Time Alarm Register (LPTAR)	32	RW	0000_0000h
5Ch	SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR)	32	RO	0000_0000h
60h	SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR)	32	RO	0000_0000h
64h	SNVS_LP Power Glitch Detector Register (LPPGDR)	32	RW	0000_0000h
68h	SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias)	32	RW	0000_0000h
6Ch - 88h	SNVS_LP Zeroizable Master Key Register (LPZMKR0 - LPZMKR7)	32	RW	0000_0000h
90h - 9Ch	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_alias - LPGPR3_alias)	32	RW	0000_0000h
100h - 10Ch	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0 - LPGPR3)	32	RW	0000_0000h
BF8h	SNVS_HP Version ID Register 1 (HPVIDR1)	32	RO	003E_0104h
BFC	SNVS_HP Version ID Register 2 (HPVIDR2)	32	RO	0600_0000h

10.8.2 SNVS_HP Lock Register (HPLR)

10.8.2.1 Offset

Register	Offset
HPLR	0h

10.8.2.2 Function

The SNVS_HP Lock Register contains lock bits for the SNVS registers. This is a privileged write register.

10.8.2.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved						MKS_SL	LPTDCR_S_L	Reserved	LPSVCR_S_L	GPR_S_L	MC_SL	LPCALB_SL	SRTC_S_L	ZMK_RSL	ZMK_WSL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.8.2.4 Fields

Field	Function
31-29	Reserved.
—	
28	Reserved.
—	
27	Reserved.
—	
26	Reserved.

Table continues on the next page...

SNVS register descriptions

Field	Function
—	
25	Reserved.
—	
24	Reserved.
—	
23-19	Reserved.
—	
18	High Assurance Counter Lock
HAC_L	When set, prevents any writes to HPHACIVR, HPHACR, and HAC_EN bit of HPCOMR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
17	HP Security Interrupt Control Register Lock
HPSICR_L	When set, prevents any writes to the HPSICR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
16	HP Security Violation Control Register Lock
HPSVCR_L	When set, prevents any writes to the HPSVCR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
15-10	Reserved.
—	
9	Master Key Select Soft Lock
MKS_SL	When set, prevents any writes to the MASTER_KEY_SEL field of the LPMKCR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
8	LP Tamper Detectors Configuration Register Soft Lock
LPTDCR_SL	When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
7	Reserved
—	
6	LP Security Violation Control Register Soft Lock
LPSVCR_SL	When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
5	General Purpose Register Soft Lock
GPR_SL	When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
4	Monotonic Counter Soft Lock

Table continues on the next page...

Field	Function
MC_SL	When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the system reset. 0b - Write access (increment) is allowed 1b - Write access (increment) is not allowed
3 LPCALB_SL	LP Calibration Soft Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
2 SRTC_SL	Secure Real Time Counter Soft Lock When set, prevents any writes to the SRTC Registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
1 ZMK_RSL	Zeroizable Master Key Read Soft Lock When set, prevents any software reads to the ZMK Registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), the ZMK and ZMK_ECC_VALUE cannot be read by software. Regardless of the bit setting, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the system reset. 0b - Read access is allowed (only in software Programming mode) 1b - Read access is not allowed
0 ZMK_WSL	Zeroizable Master Key Write Soft Lock When set, prevents any writes (software and hardware) to the ZMK registers and the ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be cleared by system reset. 0b - Write access is allowed 1b - Write access is not allowed

10.8.3 SNVS_HP Command Register (HPCOMR)

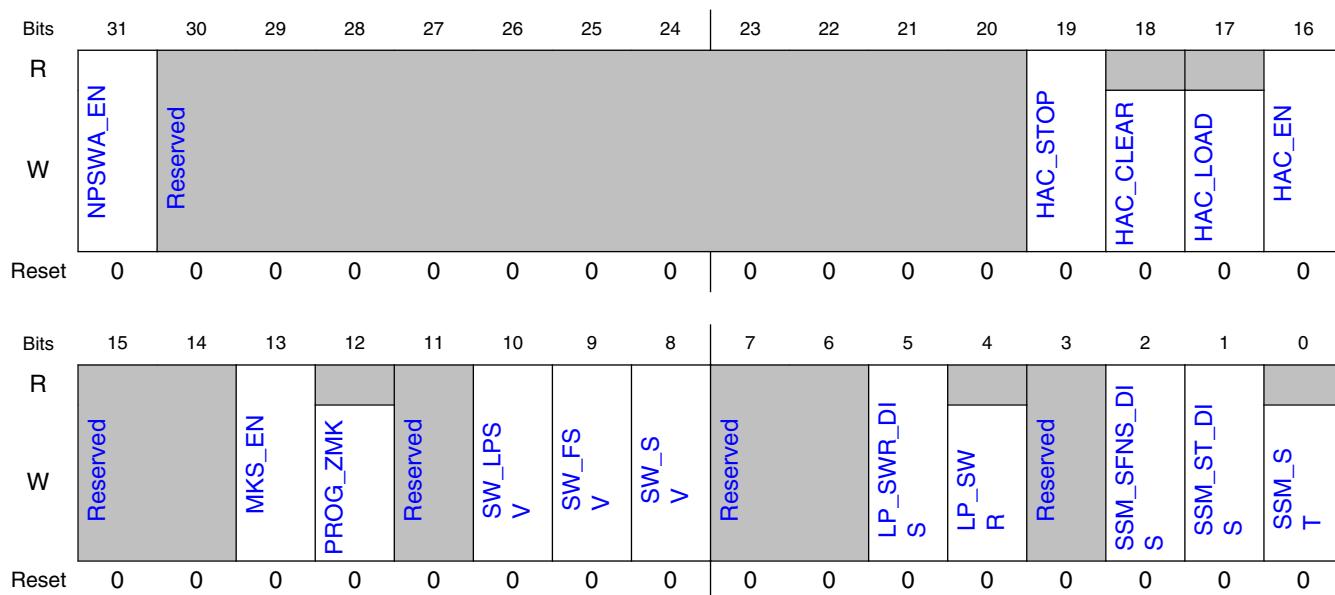
10.8.3.1 Offset

Register	Offset
HPCOMR	4h

10.8.3.2 Function

The SNVS_HP Command Register contains the command, configuration, and control bits for the SNVS block. Some fields of this register can be written to in check and soft fail states in addition to the standard write access in functional states. This is a privileged write register.

10.8.3.3 Diagram



10.8.3.4 Fields

Field	Function
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SNVS registers, including those that are privileged software read/write access only. 0 Only privileged software can access privileged registers 1 Any software can access privileged registers
30-20 —	Reserved.
19 HAC_STOP	High Assurance Counter Stop

Table continues on the next page...

Field	Function
	<p>This bit can be set only when SSM is in soft fail state. When set, it stops the high assurance counter and prevents transition to the hard fail state. This bit can be cleared in a functional or soft fail state. If the bit is cleared in the soft fail state, the high assurance counter counts down from the place where it was stopped.</p> <p>0 HAC counter can count down 1 HAC counter is stopped</p>
18 HAC_CLEAR	<p>High Assurance Counter Clear</p> <p>When set, it clears the High Assurance Counter Register. It can be cleared in a functional or soft fail state. If the HAC counter is cleared in the soft fail state, the SSM transitions to the hard fail state if high assurance counter is enabled (HAC_EN is set). This self-clearing bit is always read as zero.</p> <p>0b - No Action 1b - Clear the HAC</p>
17 HAC_LOAD	<p>High Assurance Counter Load</p> <p>When set, it loads the High Assurance Counter Register with the value of the High Assurance Counter Load Register. It can be done in a functional or soft fail state. This self-clearing bit is always read as zero.</p> <p>0b - No Action 1b - Load the HAC</p>
16 HAC_EN	<p>High Assurance Counter Enable</p> <p>This bit controls the SSM transition from the soft fail to the hard fail state. When this bit is set and software fails to stop the HAC before it expires, the SSM transitions to the hard fail state. This bit cannot be changed once HAC_L bit is set.</p> <p>0b - High Assurance Counter is disabled 1b - High Assurance Counter is enabled</p>
15-14 —	Reserved.
13 MKS_EN	<p>Master Key Select Enable</p> <p>When not set, the one time programmable (OTP) master key is selected by default. When set, the master key is selected according to the setting of the master key select field (MASTER_KEY_SEL) of LPMKCR. Once set, this bit can only be reset by the system reset.</p> <p>0b - OTP master key is selected as an SNVS master key 1b - SNVS master key is selected according to the setting of the MASTER_KEY_SEL field of LPMKCR</p>
12 PROG_ZMK	<p>Program Zeroizable Master Key</p> <p>This bit activates ZMK hardware programming mechanism. This mechanism is activated only if the ZMK is configured to the hardware programming mode and ZMK is not locked for writes. This self-clearing bit is always read as zero.</p> <p>0b - No Action 1b - Activate hardware key programming mechanism</p>
11 —	Reserved.
10 SW_LPSV	<p>LP Software Security Violation</p> <p>When set, SNVS_LP treats this bit as a security violation. The LP secure data is zeroized or invalidated according to the configuration. This security violation may result in a system security monitor transition if the LP Security Violation is enabled in the SNVS_HP Security Violation Control Register.</p>
9 SW_FSV	Software Fatal Security Violation

Table continues on the next page...

SNVS register descriptions

Field	Function
	<p>When set, the system security monitor treats this bit as a fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM:</p> <ul style="list-style-type: none"> Check State -> Soft Fail Non-Secure State -> Soft Fail Trusted State -> Soft Fail Secure State -> Soft Fail
8 SW_SV	<p>Software Security Violation</p> <p>When set, the system security monitor treats this bit as a non-fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM:</p> <ul style="list-style-type: none"> Check -> Non-Secure Trusted -> Soft Fail Secure -> Soft Fail
7-6 —	Reserved.
5 LP_SWR_DIS	<p>LP Software Reset Disable</p> <p>When set, disables the LP software reset. Once set, this bit can only be reset by the system reset.</p> <ul style="list-style-type: none"> 0b - LP software reset is enabled 1b - LP software reset is disabled
4 LP_SWR	<p>LP Software Reset</p> <p>When set to 1, most registers in the SNVS_LP section are reset, but the following registers are <i>not</i> reset by an LP software reset:</p> <ul style="list-style-type: none"> • Secure Real Time Counter • Time Alarm Register <p>This bit cannot be set when the LP_SWR_DIS bit is set. This self-clearing bit is always read as zero.</p> <ul style="list-style-type: none"> 0b - No Action 1b - Reset LP section
3 —	Reserved.
2 SSM_SFNS_DIS	<p>SSM Soft Fail to Non-Secure State Transition Disable</p> <p>When set, it disables the SSM transition from soft fail to non-secure state. Once set after the reset this bit cannot be changed</p> <ul style="list-style-type: none"> 0b - Soft Fail to Non-Secure State transition is enabled 1b - Soft Fail to Non-Secure State transition is disabled
1 SSM_ST_DIS	<p>SSM Secure to Trusted State Transition Disable</p> <p>When set, disables the SSM transition from secure to trusted state. Once set after the reset, this bit cannot be changed.</p> <ul style="list-style-type: none"> 0b - Secure to Trusted State transition is enabled 1b - Secure to Trusted State transition is disabled
0 SSM_ST	<p>SSM State Transition</p> <p>Transition state of the system security monitor. This self-clearing bit is always read as zero. This command results only in the following transitions of the SSM:</p> <p>Check State → Non-Secure (when Non-Secure Boot and not in Fab Configuration)</p>

Field	Function
	Check State --> Trusted (when Secure Boot or in Fab Configuration) Trusted State --> Secure Secure State --> Trusted (if not disabled by SSM_ST_DIS bit) Soft Fail --> Non-Secure (if not disabled by SSM_SFNS_DIS bit)

10.8.4 SNVS_HP Control Register (HPCR)

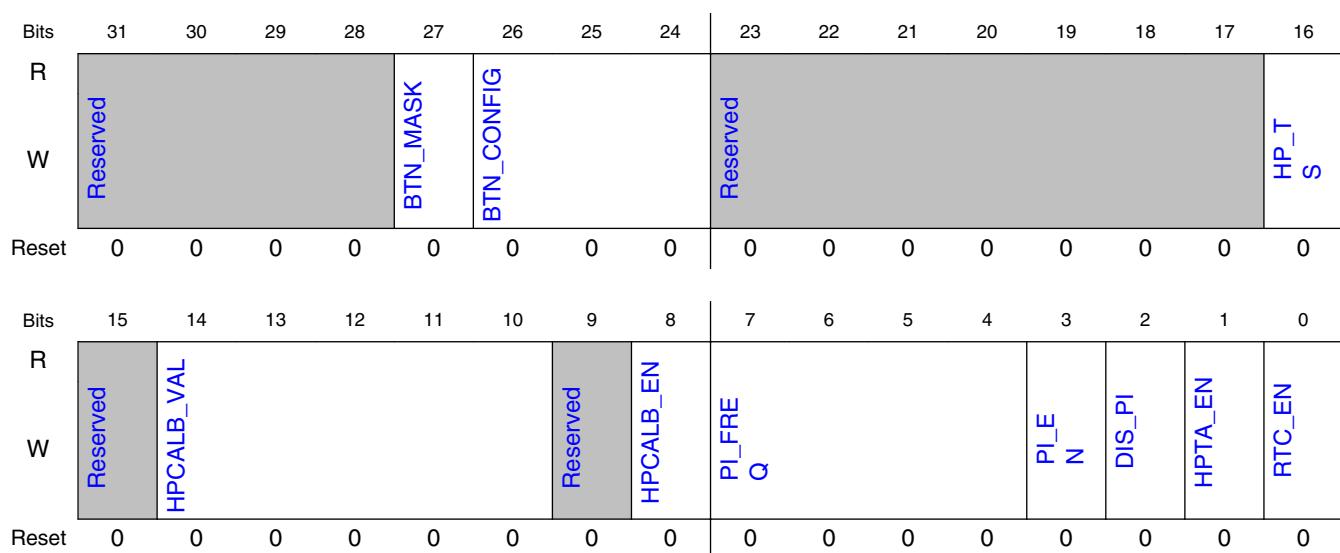
10.8.4.1 Offset

Register	Offset
HPCR	8h

10.8.4.2 Function

The SNVS_HP Control Register contains various control bits of the HP section of SNVS. This is *not* a privileged write register.

10.8.4.3 Diagram



10.8.4.4 Fields

Field	Function
31-28 —	Reserved.
27 BTN_MASK	Button interrupt mask. This bit is used to mask the ipi_snvs_btn_int_b (button) interrupt request. 0: Interrupt disabled 1: Interrupt enabled
26-24 BTN_CONFIG	Button Configuration. This field is used to configure which feature of the button (BTN) input signal constitutes "active". 000: Button signal is active high 001: Button signal is active low 010: Button signal is active on the falling edge 011: Button signal is active on the rising edge 100: Button signal is active on any edge All other patterns are reserved.
23-17 —	Reserved.
16 HP_TS	HP Time Synchronize. HP RTC should be disabled when synchronizing counter value. When set, this updates the HP Real Time Counter with the LP Real Time Counter value. This self-clearing bit is always read as zero. 0b - No Action 1b - Synchronize the HP Time Counter to the LP Time Counter
15 —	Reserved.
14-10 HPCALB_VAL	HP Calibration Value Defines signed calibration value for the HP Real Time Counter. This field can be programmed only when RTC Calibration is disabled (HPCALB_EN is not set). This is a 5-bit 2's complement value, hence the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter. 00000b - +0 counts per each 32768 ticks of the counter 00001b - +1 counts per each 32768 ticks of the counter 00010b - +2 counts per each 32768 ticks of the counter 01111b - +15 counts per each 32768 ticks of the counter 10000b - -16 counts per each 32768 ticks of the counter 10001b - -15 counts per each 32768 ticks of the counter 11110b - -2 counts per each 32768 ticks of the counter 11111b - -1 counts per each 32768 ticks of the counter
9 —	Reserved.
8 HPCALB_EN	HP Real Time Counter Calibration Enabled Indicates that the time calibration mechanism is enabled. 0b - HP Timer calibration disabled

Table continues on the next page...

Field	Function
	1b - HP Timer calibration enabled
7-4 PI_FREQ	<p>Periodic Interrupt Frequency</p> <p>Defines frequency of the periodic interrupt. The interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the HP Real Time Counter and Real Time Counter and Periodic Interrupt are both enabled (RTC_EN and PI_EN are set). It is recommended to program this field when Periodic Interrupt is disabled (PI_EN is not set). The possible frequencies are:</p> <ul style="list-style-type: none"> 0000b - - bit 0 of the HPRTCLR is selected as a source of the periodic interrupt 0001b - - bit 1 of the HPRTCLR is selected as a source of the periodic interrupt 0010b - - bit 2 of the HPRTCLR is selected as a source of the periodic interrupt 0011b - - bit 3 of the HPRTCLR is selected as a source of the periodic interrupt 0100b - - bit 4 of the HPRTCLR is selected as a source of the periodic interrupt 0101b - - bit 5 of the HPRTCLR is selected as a source of the periodic interrupt 0110b - - bit 6 of the HPRTCLR is selected as a source of the periodic interrupt 0111b - - bit 7 of the HPRTCLR is selected as a source of the periodic interrupt 1000b - - bit 8 of the HPRTCLR is selected as a source of the periodic interrupt 1001b - - bit 9 of the HPRTCLR is selected as a source of the periodic interrupt 1010b - - bit 10 of the HPRTCLR is selected as a source of the periodic interrupt 1011b - - bit 11 of the HPRTCLR is selected as a source of the periodic interrupt 1100b - - bit 12 of the HPRTCLR is selected as a source of the periodic interrupt 1101b - - bit 13 of the HPRTCLR is selected as a source of the periodic interrupt 1110b - - bit 14 of the HPRTCLR is selected as a source of the periodic interrupt 1111b - - bit 15 of the HPRTCLR is selected as a source of the periodic interrupt
3 PI_EN	<p>HP Periodic Interrupt Enable</p> <p>The periodic interrupt can be generated only if the HP Real Time Counter is enabled.</p> <ul style="list-style-type: none"> 0b - HP Periodic Interrupt is disabled 1b - HP Periodic Interrupt is enabled
2 DIS_PI	<p>Disable periodic interrupt in the functional interrupt</p> <ul style="list-style-type: none"> 0b - Periodic interrupt will trigger a functional interrupt 1b - Disable periodic interrupt in the function interrupt
1 HPTA_EN	<p>HP Time Alarm Enable</p> <p>When set, the time alarm interrupt is generated if the value in the HP Time Alarm Registers is equal to the value of the HP Real Time Counter.</p> <ul style="list-style-type: none"> 0b - HP Time Alarm Interrupt is disabled 1b - HP Time Alarm Interrupt is enabled
0 RTC_EN	<p>HP Real Time Counter Enable. This bit syncs with the 32KHz clock. It won't update with the bus clock.</p> <ul style="list-style-type: none"> 0b - RTC is disabled 1b - RTC is enabled

10.8.5 SNVS_HP Security Interrupt Control Register (HPSICR)

10.8.5.1 Offset

Register	Offset
HPSICR	Ch

10.8.5.2 Function

The HP Security Interrupt Control Register defines the SNVS security interrupt generation policy. This is a privileged write register.

10.8.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LPSVI_EN															
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								SV5_EN	SV4_EN	SV3_EN	SV2_EN	SV1_EN	SV0_EN		
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.8.5.4 Fields

Field	Function
31 LPSVI_EN	LP Security Violation Interrupt Enable This bit enables generating of the security interrupt to the host processor upon security violation signal from the LP section. 0b - LP Security Violation Interrupt is Disabled 1b - LP Security Violation Interrupt is Enabled
30-6 —	Reserved
5 SV5_EN	Security Violation 5 Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Violation 5 security violation. 0b - Security Violation 5 Interrupt is Disabled 1b - Security Violation 5 Interrupt is Enabled
4 SV4_EN	Security Violation 4 Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Violation 4 security violation.

Table continues on the next page...

Field	Function
	0b - Security Violation 4 Interrupt is Disabled 1b - Security Violation 4 Interrupt is Enabled
3 SV3_EN	Security Violation 3 Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Violation 3 security violation. 0b - Security Violation 3 Interrupt is Disabled 1b - Security Violation 3 Interrupt is Enabled
2 SV2_EN	Security Violation 2 Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Violation 2 security violation. 0b - Security Violation 2 Interrupt is Disabled 1b - Security Violation 2 Interrupt is Enabled
1 SV1_EN	Security Violation 1 Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Violation 1 security violation. 0b - Security Violation 1 Interrupt is Disabled 1b - Security Violation 1 Interrupt is Enabled
0 SV0_EN	Security Violation 0 Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Violation 0 security violation. 0b - Security Violation 0 Interrupt is Disabled 1b - Security Violation 0 Interrupt is Enabled

10.8.6 SNVS_HP Security Violation Control Register (HPSVCR)

10.8.6.1 Offset

Register	Offset
HPSVCR	10h

10.8.6.2 Function

The HP Security Violation Control Register defines types for each security violation input. This is a privileged write register.

10.8.6.3 Diagram

Bits	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	LPSV_CFG																
W	Reserved																
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	Reserved																
W	SV5_CFG																
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

10.8.6.4 Fields

Field	Function
31-30 LPSV_CFG	LP Security Violation Configuration This field configures the LP security violation source. 00b - LP security violation is disabled 01b - LP security violation is a non-fatal violation 1xb - LP security violation is a fatal violation
29-7 —	Reserved.
6-5 SV5_CFG	Security Violation 5 Security Violation Configuration This field configures the Security Violation 5 Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Violation 5 security violation. 00b - Security Violation 5 is disabled 01b - Security Violation 5 is a non-fatal violation 1xb - Security Violation 5 is a fatal violation
4 SV4_CFG	Security Violation 4 Security Violation Configuration This field configures the Security Violation 4 Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Violation 4 security violation. 0b - Security Violation 4 is a non-fatal violation 1b - Security Violation 4 is a fatal violation
3 SV3_CFG	Security Violation 3 Security Violation Configuration This field configures the Security Violation 3 Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Violation 3 security violation. 0b - Security Violation 3 is a non-fatal violation 1b - Security Violation 3 is a fatal violation
2	Security Violation 2 Security Violation Configuration

Table continues on the next page...

Field	Function
SV2_CFG	This field configures the Security Violation 2 Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Violation 2 security violation. 0b - Security Violation 2 is a non-fatal violation 1b - Security Violation 2 is a fatal violation
1 SV1_CFG	Security Violation 1 Security Violation Configuration This field configures the Security Violation 1 Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Violation 1 security violation. 0b - Security Violation 1 is a non-fatal violation 1b - Security Violation 1 is a fatal violation
0 SV0_CFG	Security Violation 0 Security Violation Configuration This field configures the Security Violation 0 Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Violation 0 security violation. 0b - Security Violation 0 is a non-fatal violation 1b - Security Violation 0 is a fatal violation

10.8.7 SNVS_HP Status Register (HPSR)

10.8.7.1 Offset

Register	Offset
HPSR	14h

10.8.7.2 Function

The HP Status Register reflects the internal state of the SNVS. This is *not* a privileged write register.

10.8.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ZMK_ZERO	Reserved			OTPMK_ZERO	Reserved		OTPMK_SYNDROME								
W																
Reset	1 ¹	0	0	0	0 ²	0	0	0 ³	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SECURITY_CONFIG				SSM_STAT_E				BI	BTN	Reserved	LPDIS	Reserved	PI		
W									W1C	0	0	0	0	W1C	W1C	
Reset	0 ⁴	0	1	1	0	0	0	0	0	0	0	0 ⁵	0	0	0	0

1. This bit will reset to 1 on LP Section POR. If the LP section is powered by a coin cell battery, once the ZMK has been programmed with a nonzero value this bit will reset to 0 on HP Section PORs.
2. OTPMK_ZERO will normally reset to 0. This will reset to 1 only if the OTPMK has not been programmed in the fuse bank.
3. OTPMK_SYNDROME will normally reset to 000h. This will reset to nonzero only if the OTPMK has an error in one or more bits.
4. SYS_SECURITY_CFG will normally reset to 1011b in a chip in the field. It will reset to X000b only in a chip in the Fabrication facility. After the Fab-programmable fuses have been blown at the Fabrication facility SYS_SECURITY_CFG will reset to X001b. The OEM is expected to blow the OEM-programmable fuses, after which SYS_SECURITY_CFG will reset to 1011b.
5. The value of Low Power Disable is determined by the *no_battery* input signal to SNVS.

10.8.7.4 Fields

Field	Function
31 ZMK_ZERO	Zeroizable Master Key is Equal to Zero. When set, this bit triggers "bad key" violation if the ZMK is selected for use. 0b - The ZMK is not zero. 1b - The ZMK is zero.
30-28 —	Reserved.

Table continues on the next page...

Field	Function
27 OTPMK_ZERO	One Time Programmable Master Key is Equal to Zero. When set, this bit always triggers "bad key" violation. 0b - The OTPMK is not zero. 1b - The OTPMK is zero.
26-25 —	Reserved.
24-16 OTPMK_SYNDROME	One Time Programmable Master Key Syndrome In the case of a single-bit error, the eight lower bits of this value indicate the bit number of error location. For example, syndrome word 10010110 indicates that key bit 150 (lsb=0) has an error. The ninth bit of the syndrome word checks parity of the whole key value. This bit is 1 when an odd number of errors has occurred and it is 0 when the number of errors is even. For example, if one of the eight bits indicates a failure and the ninth bit is zero then the number of errors in the one time programmable master key is at least 2 and it cannot be corrected. When one of the syndrome bits is set, the bad key violation is always generated.
15-12 SECURITY_CONFIG	Security Configuration This field reflects the settings of the sys_secure_boot input and the three security configuration inputs to SNVS. These four inputs are interpreted as follows. sys_secure_boot : sys_security_cfg[2:0] 0000, 1000b - FAB configuration. The SNVS SSM can be transitioned from CHECK SECURE, regardless of the state of the sys_secure_boot input. 0001, 0010, 0011b - OPEN configuration. The SNVS SSM transition from CHECK state to TRUSTED state is blocked. CHECK state can be transitioned only to NONSECURE state. 1010, 1001, 1011b - CLOSED configuration. The SNVS SSM transition from CHECK state to TRUSTED state is permitted. The SSM can be transitioned from CHECK state to either TRUSTED state or NONSECURE state, as selected by the boot software. x1xxb - FIELD RETURN configuration. The SNVS SSM behaves the same as in the FAB, OPEN or CLOSED configuration (depending on the value of the sys_secure_boot and sys_security_cfg[1:0]inputs), but in Field Return Configuration the Master Key (used for cryptographic blobs) will be forced to all 0s, the test value.
11-8 SSM_STATE	System Security Monitor State This field contains the encoded state of the SSM's state machine. The encoding of the possible states are: 0000b - Init 0001b - Hard Fail 0011b - Soft Fail 1000b - Init Intermediate (transition state between Init and Check - SSM stays in this state only one clock cycle) 1001b - Check 1011b - Non-Secure 1101b - Trusted 1111b - Secure
7 BI	Button Interrupt Signal ipi_snvs_btn_int_b was asserted.
6 BTN	Button Value of the BTN input. This is the external button used for PMIC control. 0: BTN not pressed 1: BTN pressed
5	Reserved.

Table continues on the next page...

SNVS register descriptions

Field	Function
—	
4	Low Power Disable
LPDIS	If 1, the low power section has been disabled by means of an input signal to SNVS.
3-2	Reserved.
—	
1	Periodic Interrupt
PI	Indicates that periodic interrupt has occurred since this bit was last cleared. 0b - No periodic interrupt occurred. 1b - A periodic interrupt occurred.
0	HP Time Alarm
HPTA	Indicates that the HP Time Alarm has occurred since this bit was last cleared. 0b - No time alarm interrupt occurred. 1b - A time alarm interrupt occurred.

10.8.8 SNVS_HP Security Violation Status Register (HPSVSR)

10.8.8.1 Offset

Register	Offset
HPSVSR	18h

10.8.8.2 Function

The HP Security Violation Status Register reflects the HP domain security violation records. Write a 1 to Security Violation 5-0 to clear the corresponding security violation detection flag. Note that this does not automatically clear the security violation signal that is connected to the input, so the security violation may immediately be detected again. This is a privileged write register.

10.8.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
R	LP_SEC_VIO	Reserved				ZMK_ECC_FAIL	Reserved				ZMK_SYNDROME							
W					W1C	0	0	0	0	0	0	0	0	0	0	0		
Reset	1 ¹	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R	SW_LPS_V	SW_FS_V	SW_S_V	Reserved				SV5	SV4	SV3	SV2	SV1	SV0					
W	0	0	0	0	0	0	0	W1C	W1C	W1C	W1C	W1C	W1C	0	0	0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

1. This bit will reset to 1 on LP Section POR because the Power Glitch Detector will indicate that an LP section power glitch has occurred. If the LP section is powered by a coin cell battery, LP Security Violation will normally reset to 0 if no LP POR has occurred since the Power Glitch Detector was written. If a POR occurred in the LP section, at HP POR the boot ROM firmware will normally detect this and will then re-initialize the LP section and reset LP Security Violation to 0.

10.8.8.4 Fields

Field	Function
31	LP Security Violation
LP_SEC_VIO	A security violation was detected in the SNVS low power section.
30-28	Reserved.
—	
27	Zeroizable Master Key Error Correcting Code Check Failure
ZMK_ECC_FAIL	When set, this bit triggers a bad key violation to the SSM and a security violation to the SNVS_LP section, which clears security sensitive data. Writing a one to this bit clears the record of this failure and also clears this register's ZMK_SYNDROME field. 0b - ZMK ECC Failure was not detected. 1b - ZMK ECC Failure was detected.
26-25	Reserved.
—	
24-16	Zeroizable Master Key Syndrome

Table continues on the next page...

SNVS register descriptions

Field	Function
ZMK_SYNDROME	The ZMK syndrome indicates the single-bit error location and parity for the ZMK register. It operates similar to the OTPMK syndrome. This value is set and locked when a ZMK ECC failure is detected. It is cleared by writing one into ZMK_ECC_FAIL bit.
15 SW_LPSV	LP Software Security Violation This bit is a read-only copy of the SW_LPSV bit in the HP Command Register.
14 SW_FSV	Software Fatal Security Violation This bit is a read-only copy of the SW_FSV bit in the HP Command Register.
13 SW_SV	Software Security Violation This bit is a read-only copy of the SW_SV bit in the HP Command Register.
12-6 —	Reserved.
5 SV5	Security Violation 5 security violation was detected. 0b - No Security Violation 5 security violation was detected. 1b - Security Violation 5 security violation was detected.
4 SV4	Security Violation 4 security violation was detected. 0b - No Security Violation 4 security violation was detected. 1b - Security Violation 4 security violation was detected.
3 SV3	Security Violation 3 security violation was detected. 0b - No Security Violation 3 security violation was detected. 1b - Security Violation 3 security violation was detected.
2 SV2	Security Violation 2 security violation was detected. 0b - No Security Violation 2 security violation was detected. 1b - Security Violation 2 security violation was detected.
1 SV1	Security Violation 1 security violation was detected. 0b - No Security Violation 1 security violation was detected. 1b - Security Violation 1 security violation was detected.
0 SV0	Security Violation 0 security violation was detected. 0b - No Security Violation 0 security violation was detected. 1b - Security Violation 0 security violation was detected.

10.8.9 SNVS_HP High Assurance Counter IV Register (HPA CIVR)

10.8.9.1 Offset

Register	Offset
HPHACIVR	1Ch

10.8.9.2 Function

The SNVS_HP High Assurance Counter IV Register contains the initial value for the high assurance counter. This is a privileged write register.

10.8.9.3 Diagram

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

10.8.9.4 Fields

Field	Function
31-0	High Assurance Counter Initial Value
HAC_COUNTE R_IV	This register is used to set the starting count value to the high assurance counter. This register cannot be programmed when HAC_L bit is set.

10.8.10 SNVS_HP High Assurance Counter Register (PHHACR)

10.8.10.1 Offset

Register	Offset
PHHACR	20h

10.8.10.2 Function

The SNVS_HP High Assurance Counter Register contains the value of the high assurance counter. The high assurance counter is a delay introduced before the system security monitor transitions from soft fail to hard fail state if this transition is enabled. This is a non-privileged read-only register.

10.8.10.3 Diagram

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

10.8.10.4 Fields

Field	Function
31-0 HAC_COUNTE R	<p>High Assurance Counter</p> <p>When the HAC_EN bit is set and the SSM is in the soft fail state, this counter starts to count down with the system clock. When the counter reaches zero, the SSM transitions to the Hard Fail State.</p> <ul style="list-style-type: none"> When HAC_STOP bit is set, the HAC Counter is stopped. When HAC_CLEAR bit is set, the HAC Counter is cleared. When HAC_LOAD bit is set, the HAC Counter is loaded with the value of the HPHACIVR.

10.8.11 SNVS_HP Real Time Counter MSB Register (HPRTCMR)

10.8.11.1 Offset

Register	Offset
HPRTCMR	24h

10.8.11.2 Function

The SNVS_HP Real Time Counter MSB register contains the 15 most-significant bits of the HP Real Time Counter. This is *not* a privileged write register.

10.8.11.3 Diagram

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

10.8.11.4 Fields

Field	Function
31-15 —	Reserved
14-0 RTC	The most-significant 15 bits of the RTC. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

10.8.12 SNVS_HP Real Time Counter LSB Register (HPRTCLR)

10.8.12.1 Offset

Register	Offset
HPRTCLR	28h

10.8.12.2 Function

The SNVS_HP Real Time Counter LSB register contains the 32 least-significant bits of the HP real time counter. This is *not* a privileged write register.

10.8.12.3 Diagram

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

10.8.12.4 Fields

Field	Function
31-0	HP Real Time Counter
RTC	least-significant 32 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

10.8.13 SNVS_HP Time Alarm MSB Register (HPTAMR)

10.8.13.1 Offset

Register	Offset
HPTAMR	2Ch

10.8.13.2 Function

The SNVS_HP Time Alarm MSB register contains the most-significant bits of the SNVS_HP Time Alarm value. This is *not* a privileged write register.

10.8.13.3 Diagram

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

10.8.13.4 Fields

Field	Function
31-15	Reserved.
—	
14-0 HPTA_MS	HP Time Alarm, most-significant 15 bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

10.8.14 SNVS_HP Time Alarm LSB Register (HPTALR)

10.8.14.1 Offset

Register	Offset
HPTALR	30h

10.8.14.2 Function

The SNVS_HP Time Alarm LSB register contains the 32 least-significant bits of the SNVS_HP Time Alarm value. This is *not* a privileged write register.

10.8.14.3 Diagram

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

10.8.14.4 Fields

Field	Function
31-0	HP Time Alarm, 32 least-significant bits.
HPTA_LS	This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

10.8.15 SNVS_LP Lock Register (LPLR)

10.8.15.1 Offset

Register	Offset
LPLR	34h

10.8.15.2 Function

The SNVS_LP Lock Register contains lock bits for the SNVS_LP registers. This is a privileged write register.

10.8.15.3 Diagram

Bits	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	Reserved									Reserved							
W																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R	Reserved									Reserved	LPSVCR_HL	GPR_H_L	MC_HL	LPCALB_HL	SRTC_H_L	ZMK_RHL	ZMK_WHL
W										0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

10.8.15.4 Fields

Field	Function
31-29	Reserved.
—	
28	Reserved.
—	
27	Reserved.
—	
26	Reserved.
—	
25	Reserved.
—	
24	Reserved.
—	
23-10	Reserved.
—	
9	Master Key Select Hard Lock

Table continues on the next page...

SNVS register descriptions

Field	Function
MKS_HL	When set, prevents any writes to the MASTER_KEY_SEL field of the LP Master Key Control Register. Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.
8 LPTDCR_HL	LP Tamper Detectors Configuration Register Hard Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.
7 —	Reserved.
6 LPSVCR_HL	LP Security Violation Control Register Hard Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.
5 GPR_HL	General Purpose Register Hard Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.
4 MC_HL	Monotonic Counter Hard Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0b - Write access (increment) is allowed. 1b - Write access (increment) is not allowed.
3 LPCALB_HL	LP Calibration Hard Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.
2 SRTC_HL	Secure Real Time Counter Hard Lock When set, prevents any writes to the SRTC registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.
1 ZMK_RHL	Zeroizable Master Key Read Hard Lock When set, prevents any software reads to the ZMK registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), software cannot read the ZMK or ZMK_ECC_VALUE. Regardless of the setting of this bit, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the LP POR. 0b - Read access is allowed (only in software programming mode). 1b - Read access is not allowed.
0 ZMK_WHL	Zeroizable Master Key Write Hard Lock When set, prevents any writes (software and hardware) to the ZMK registers and ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be reset by the LP POR. 0b - Write access is allowed. 1b - Write access is not allowed.

10.8.16 SNVS_LP Control Register (LPCR)

10.8.16.1 Offset

Register	Offset
LPCR	38h

10.8.16.2 Function

The SNVS_LP Control Register contains various control bits of the LP section of SNVS. This is a privileged write register.

10.8.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							GPR_Z_DS	PK_OVERRIDE	PK_EN	ON_TIME	DEBOUNCE				BTN_PRESS_TIM_E
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		LPCALB_VAL						PWR_GLITCH_EN	TOP	DP_EN	SRTC_INV_EN	LPWUI_EN	MC_ENV	LPTA_EN	SRTC_EN_V
W	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

10.8.16.4 Fields

Field	Function
31-25 —	Reserved.
24 GPR_Z_DIS	General Purpose Registers Zeroization Disable. 1 = Disable zeroization of the GPR registers when a tamper event occurs. 0 = Zeroize the GPR registers when a tamper event occurs. (Default)
23 PK_OVERRIDE	PMIC On Request Override The value written to PK_OVERRIDE will be asserted on output signal snvs_lp_pk_override. That signal is used to override the IOMUX control for the PMIC I/O pad.
22 PK_EN	PMIC On Request Enable The value written to PK_EN will be asserted on output signal snvs_lp_pk_en. That signal is used to turn off the pullup/pulldown circuitry in the PMIC I/O pad.
21-20 ON_TIME	The ON_TIME field is used to configure the period of time after BTN is asserted before pmic_en_b is asserted to turn on the SoC power. 00: 500msec off->on transition time 01: 50msec off->on transition time 10: 100msec off->on transition time 11: 0msec off->on transition time
19-18 DEBOUNCE	This field configures the amount of debounce time for the BTN input signal. 00: 50msec debounce 01: 100msec debounce 10: 500msec debounce 11: 0msec debounce
17-16 BTN_PRESS_TIME	This field configures the button press time out values for the PMIC Logic. 00 : 5 secs 01 : 10 secs 10 : 15 secs 11 : long press disabled (pmic_en_b will not be asserted regardless of how long BTN is asserted)
15 —	Reserved.
14-10 LPCALB_VAL	LP Calibration Value Defines signed calibration value for SRTC. This field can be programmed only when SRTC calibration is disabled and not locked, i.e. when LPCALB_EN, LPCALB_SL, and LPCALB_HL bits are not set. This is a 5-bit 2's complement value. Hence, the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter clock 00000b - +0 counts per each 32768 ticks of the counter clock 00001b - +1 counts per each 32768 ticks of the counter clock 00010b - +2 counts per each 32768 ticks of the counter clock 01111b - +15 counts per each 32768 ticks of the counter clock 10000b - -16 counts per each 32768 ticks of the counter clock 10001b - -15 counts per each 32768 ticks of the counter clock

Table continues on the next page...

Field	Function
	11110b - -2 counts per each 32768 ticks of the counter clock 11111b - -1 counts per each 32768 ticks of the counter clock
9 —	Reserved.
8 LPCALB_EN	LP Calibration Enable When set, enables the SRTC calibration mechanism. This bit cannot be changed once LPCALB_SL or LPCALB_HL bit is set. 0b - SRTC Time calibration is disabled. 1b - SRTC Time calibration is enabled.
7 PWR_GLITCH_EN	Power Glitch Enable By default the detection of a power glitch does not cause the pmic_en_b signal to be asserted. Setting the Power Glitch Enable bit to 1 enables the power glitch event for the PMIC. 0 - disabled 1 - enabled
6 TOP	Turn off System Power Asserting this bit causes a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0b - Leave system power on. 1b - Turn off system power.
5 DP_EN	Dumb PMIC Enabled When set, software can control the system power. When cleared, the system requires a Smart PMIC to automatically turn power off. 0b - Smart PMIC enabled. 1b - Dumb PMIC enabled.
4 SRTC_INV_EN	If this bit is 1, in the case of a security violation the SRTC stops counting and the SRTC is invalidated (SRTC_ENV bit is cleared). This is intended to allow software to read the time at which the security violation occurred. This field cannot be changed once SRTC_SL or SRTC_HL bit is set. 0b - SRTC stays valid in the case of security violation. 1b - SRTC is invalidated in the case of security violation.
3 LPWUI_EN	LP Wake-Up Interrupt Enable This interrupt line should be connected to the external pin and is intended to inform the external chip about an SNVS_LP event (tamper event, MC rollover, SRTC rollover, or time alarm). This wake-up signal can be asserted only when the chip (HP section) is powered down, and the LP section is isolated. 0 LP wake-up interrupt is disabled. 1 LP wake-up interrupt is enabled.
2 MC_ENV	Monotonic Counter Enabled and Valid When set, the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). Once MC_SL or MC_HL bit is set this bit can be changed only by LP software reset or LP POR. 0b - MC is disabled or invalid. 1b - MC is enabled and valid.
1 LPTA_EN	LP Time Alarm Enable When set, the SNVS functional interrupt is asserted if the LP Time Alarm Register is equal to the 32 MSBs of the secure real time counter. 0b - LP time alarm interrupt is disabled. 1b - LP time alarm interrupt is enabled.

Table continues on the next page...

SNVS register descriptions

Field	Function
0 SRTC_ENV	Secure Real Time Counter Enabled and Valid When set, the SRTC becomes operational. This bit cannot be changed once SRTC_SL or SRTC_HL bit is set. 0b - SRTC is disabled or invalid. 1b - SRTC is enabled and valid.

10.8.17 SNVS_LP Master Key Control Register (LPMKCR)

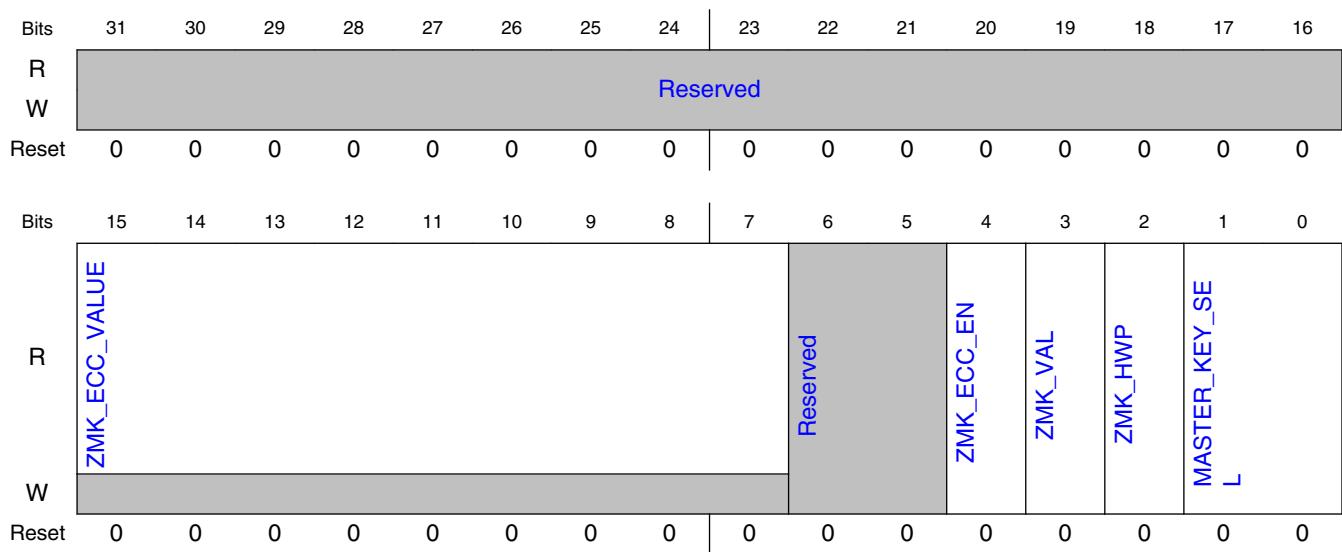
10.8.17.1 Offset

Register	Offset
LPMKCR	3Ch

10.8.17.2 Function

The SNVS_LP Master Key Control Register contains the master keys configuration. This is a privileged write register.

10.8.17.3 Diagram



10.8.17.4 Fields

Field	Function
31-16 —	Reserved.
15-7 ZMK_ECC_VAL UE	<p>Zeroizable Master Key Error Correcting Code Value</p> <p>This field is automatically calculated and set when one is written into ZMK_ECC_EN bit of this register. This field cannot be programmed by software. It keeps the ECC value of the zeroizable master key, which allows checking that ZMK has not been corrupted/changed with time.</p> <p>Note that this ZMK ECC code is equivalent to the ECC bits encoded into the OTPMK value but for the ZMK, the ECC value is kept separate from the ZMK value. Read restrictions similar to the ZMK Registers are applied to this field (see ZMK programming guidelines).</p>
6-5 —	Reserved.
4 ZMK_ECC_EN	<p>Zeroizable Master Key Error Correcting Code Check Enable</p> <p>Writing one to this field automatically calculates and sets the ZMK ECC value in the ZMK_ECC_VALUE field of this register. When both ZMK value is valid (ZMK_VAL is set) and ZMK ECC check is enabled (ZMK_ECC_EN is set), the ZMK value is continuously checked for the valid ECC word. If the ZMK ECC word calculated every clock cycle does not match the one recorded in this register, the ZMK ECC Check Fail Violation is generated. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set.</p> <p>0b - ZMK ECC check is disabled. 1b - ZMK ECC check is enabled.</p>
3 ZMK_VAL	<p>Zeroizable Master Key Valid</p> <p>When set, the ZMK value can be selected by the master key control block for use by cryptographic modules. In hardware programming mode, hardware sets this bit when the ZMK provisioning is complete. In software programming mode, software should set this bit. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set.</p> <p>0b - ZMK is not valid. 1b - ZMK is valid.</p>
2 ZMK_HWP	<p>Zeroizable Master Key hardware Programming mode</p> <p>When set, only the hardware key programming mechanism can set the ZMK and software cannot read it. When not set, the ZMK can be programmed only by software. See ZMK programming guidelines for details. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set.</p> <p>0b - ZMK is in the software programming mode. 1b - ZMK is in the hardware programming mode.</p>
1-0 MASTER_KEY_SEL	<p>Master Key Select</p> <p>These bits select the SNVS Master Key output when Master Key Select bits are enabled by MKS_EN bit in the HPCOMR. When MKS_EN bit is not set, the one time programmable master key is selected by default. This field cannot be programmed when the MKS_SL or the hard lock bit is set.</p> <p>0xb - Select one time programmable master key. 10b - Select zeroizable master key when MKS_EN bit is set . 11b - Select combined master key when MKS_EN bit is set .</p>

10.8.18 SNVS_LP Security Violation Control Register (LPSVCR)

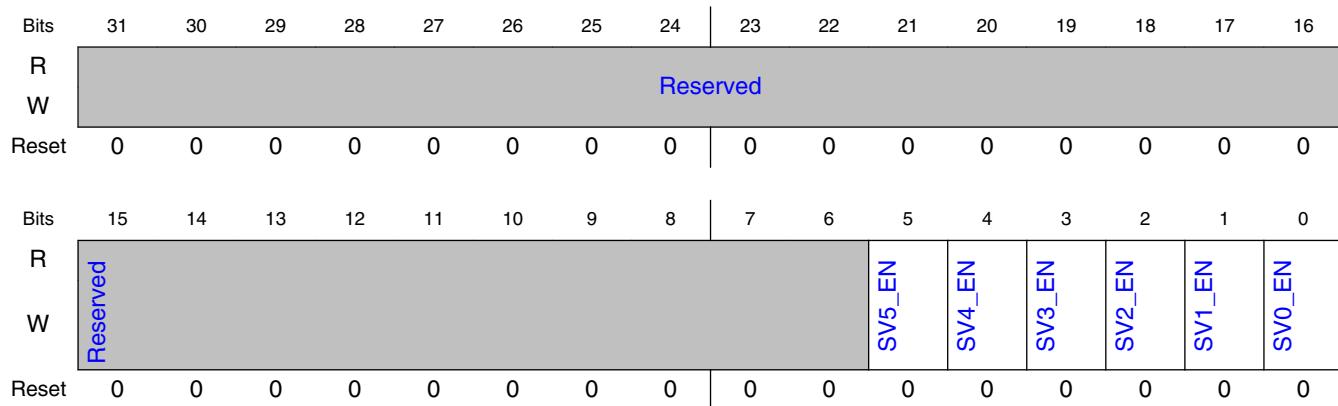
10.8.18.1 Offset

Register	Offset
LPSVCR	40h

10.8.18.2 Function

The LP Security Violation Control Register configures security violation inputs. This register cannot be programmed when the LPSVCR Lock bit is set. Note that configurations of the security violation inputs in the HP section (HPSVCR) and LP section (LPSVCR Register) are independent and have different functionality. This is a privileged write register.

10.8.18.3 Diagram



10.8.18.4 Fields

Field	Function
31-6	Reserved.
—	
5	Security Violation 5 Enable
SV5_EN	

Table continues on the next page...

Field	Function
	This bit enables Security Violation 5 Input. When set, a Security Violation 5 causes an LP security violation, which clears LP sensitive data. 0b - Security Violation 5 is disabled in the LP domain. 1b - Security Violation 5 is enabled in the LP domain.
4 SV4_EN	Security Violation 4 Enable This bit enables Security Violation 4 Input. When set, a Security Violation 4 causes an LP security violation, which clears LP sensitive data. 0b - Security Violation 4 is disabled in the LP domain. 1b - Security Violation 4 is enabled in the LP domain.
3 SV3_EN	Security Violation 3 Enable This bit enables Security Violation 3 Input. When set, a Security Violation 3 causes an LP security violation, which clears LP sensitive data. 0b - Security Violation 3 is disabled in the LP domain. 1b - Security Violation 3 is enabled in the LP domain.
2 SV2_EN	Security Violation 2 Enable This bit enables Security Violation 2 Input. When set, a Security Violation 2 causes an LP security violation, which clears LP sensitive data. 0b - Security Violation 2 is disabled in the LP domain. 1b - Security Violation 2 is enabled in the LP domain.
1 SV1_EN	Security Violation 1 Enable This bit enables Security Violation 1 Input. When set, a Security Violation 1 causes an LP security violation, which clears LP sensitive data. 0b - Security Violation 1 is disabled in the LP domain. 1b - Security Violation 1 is enabled in the LP domain.
0 SV0_EN	Security Violation 0 Enable This bit enables Security Violation 0 Input. When set, a Security Violation 0 causes an LP security violation, which clears LP sensitive data. 0b - Security Violation 0 is disabled in the LP domain. 1b - Security Violation 0 is enabled in the LP domain.

10.8.19 SNVS_LP Tamper Detectors Configuration Register (LPTDCR)

10.8.19.1 Offset

Register	Offset
LPTDCR	48h

10.8.19.2 Function

The SNVS_LP Tamper Detectors Configuration Register is used to configure analog and digital tamper detector sources. This register cannot be programmed when LPTDCR is locked for write. This is a privileged write register.

10.8.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				OSCB	Reserved	Reserved	Reserved				Reserved	Reserved	Reserved		
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POR_OBSERV	PFD_OBSERV	Reserved	Reserved	ET1P	Reserved	ET1_EN	Reserved	Reserved						MCR_EN	SRTCEN
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.8.19.4 Fields

Field	Function
31-29 —	Reserved.
28 OSCB	Oscillator Bypass When OSCB=1 the osc_bypass signal is asserted. That signal asks SoC logic external to SNVS to bypass the SoC's normal SRTC clock source and drive the SRTC clock from an alternate source. 0b - Normal SRTC clock oscillator not bypassed. 1b - Normal SRTC clock oscillator bypassed. Alternate clock can drive the SRTC clock source.
27 —	Reserved.
26-24 —	Reserved.
23 —	Reserved.
22-20 —	Reserved.

Table continues on the next page...

Field	Function
—	
19	Reserved.
—	
18-16	Reserved.
—	
15	Power On Reset (POR) Observability Flop
POR_OBSERV	The asynchronous reset input of this flop is connected directly to the output of the POR analog circuitry (external to the SNVS). This flop can be used to detect brown-out voltage of the POR circuitry.
14	System Power Fail Detector (PFD) Observability Flop
PFD_OBSERV	The asynchronous reset input of this flop is connected directly to the inverted output of the PFD analog circuitry (external to the SNVS block). This flop can be used to detect brown-out voltage of the PFD circuitry.
13	Reserved.
—	
12	Reserved.
—	
11	External Tampering 1 Polarity
ET1P	This bit is used to determine the polarity of external tamper 1. 0b - External tamper 1 is active low. 1b - External tamper 1 is active high.
10	Reserved.
—	
9	External Tampering 1 Enable
ET1_EN	When set, external tampering 1 detection generates an LP security violation. 0b - External tamper 1 is disabled. 1b - External tamper 1 is enabled.
8-3	Reserved.
—	
2	MC Rollover Enable
MCR_EN	When set, an MC Rollover event generates an LP security violation. 0b - MC rollover is disabled. 1b - MC rollover is enabled.
1	SRTC Rollover Enable
SRTCR_EN	When set, an SRTC rollover event generates an LP security violation. 0b - SRTC rollover is disabled. 1b - SRTC rollover is enabled.
0	Reserved.
—	

10.8.20 SNVS_LP Status Register (LPSR)

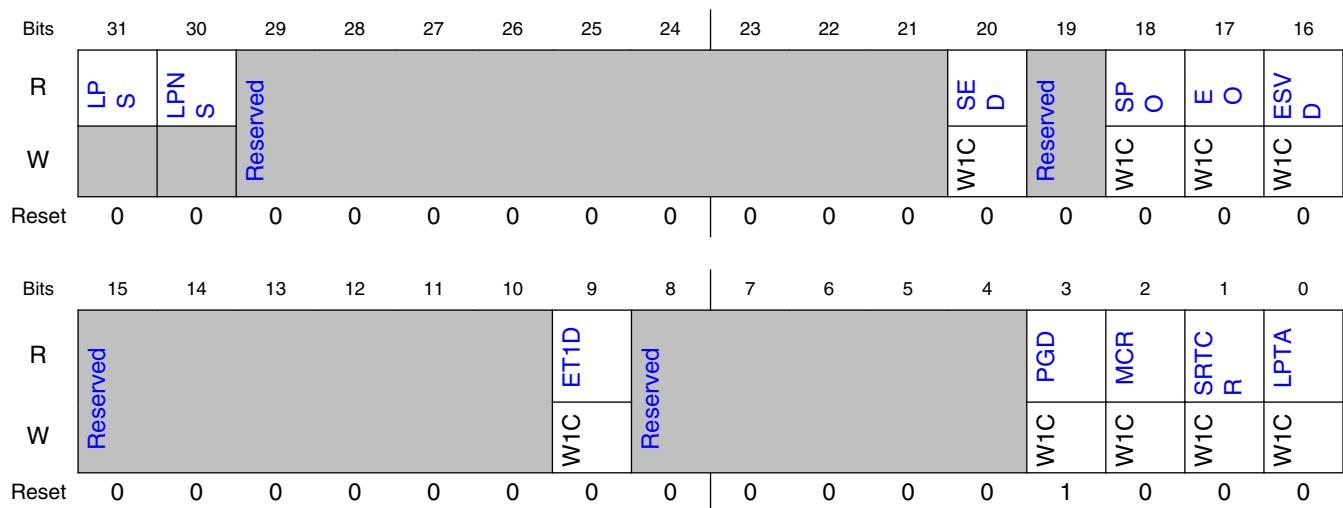
10.8.20.1 Offset

Register	Offset
LPSR	4Ch

10.8.20.2 Function

The SNVS_LP Status Register reflects the internal state and behavior of the SNVS_LP. This is a privileged write register.

10.8.20.3 Diagram



10.8.20.4 Fields

Field	Function
31 LPS	<p>LP Section is Secured</p> <p>Indicates that the LP section is provisioned/programmed in the secure or trusted state. The first write to the LP registers in secure or trusted state sets this bit. This bit can never be set together with the LPN bit. When set the SNVS_LP section cannot be programmed and ZMK cannot be read in the non-secure state of the SSM.</p> <p>0b - LP section was not programmed in secure or trusted state.</p>

Table continues on the next page...

Field	Function
	1b - LP section was programmed in secure or trusted state.
30 LPNS	<p>LP Section is Non-Secured</p> <p>Indicates that LP section was provisioned/programmed in the non-secure state. The first successful write to the LP Registers in non-secure state sets this bit. This bit can never be set together with the LPS bit. When set, the entire SNVS_LP section (all LP registers) are cleared upon an SSM transition from check to trusted state.</p> <p>0b - LP section was not programmed in the non-secure state. 1b - LP section was programmed in the non-secure state.</p>
29-21 —	Reserved.
20 SED	<p>Scan Exit Detected</p> <p>0b - Scan exit was not detected. 1b - Scan exit was detected.</p>
19 —	Reserved.
18 SPO	<p>Set Power Off</p> <p>The SPO bit is set when the power button is pressed longer than the configured debounce time. Writing to the SPO bit will clear the set_pwr_off_irq interrupt.</p> <p>0b - Set Power Off was not detected. 1b - Set Power Off was detected.</p>
17 EO	<p>Emergency Off</p> <p>This bit is set when a power off is requested.</p> <p>0b - Emergency off was not detected. 1b - Emergency off was detected.</p>
16 ESVD	<p>External Security Violation Detected</p> <p>Indicates that a security violation is detected on one of the HP security violation ports. The record of the port on which the violation has occurred can be found in the HP Security Violation Status Register.</p> <p>0b - No external security violation. 1b - External security violation is detected.</p>
15-10 —	Reserved.
9 ET1D	<p>External Tampering 1 Detected</p> <p>0b - External tampering 1 not detected. 1b - External tampering 1 detected.</p>
8-4 —	Reserved.
3 PGD	<p>Power Supply Glitch Detected</p> <p>0 No power supply glitch. 1 Power supply glitch is detected.</p>
2 MCR	<p>Monotonic Counter Rollover</p> <p>0b - MC has not reached its maximum value. 1b - MC has reached its maximum value.</p>
1	Secure Real Time Counter Rollover

Table continues on the next page...

SNVS register descriptions

Field	Function
SRTCR	0b - SRTC has not reached its maximum value. 1b - SRTC has reached its maximum value.
0 LPTA	LP Time Alarm 0b - No time alarm interrupt occurred. 1b - A time alarm interrupt occurred.

10.8.21 SNVS_LP Secure Real Time Counter MSB Register (LPSRTCMR)

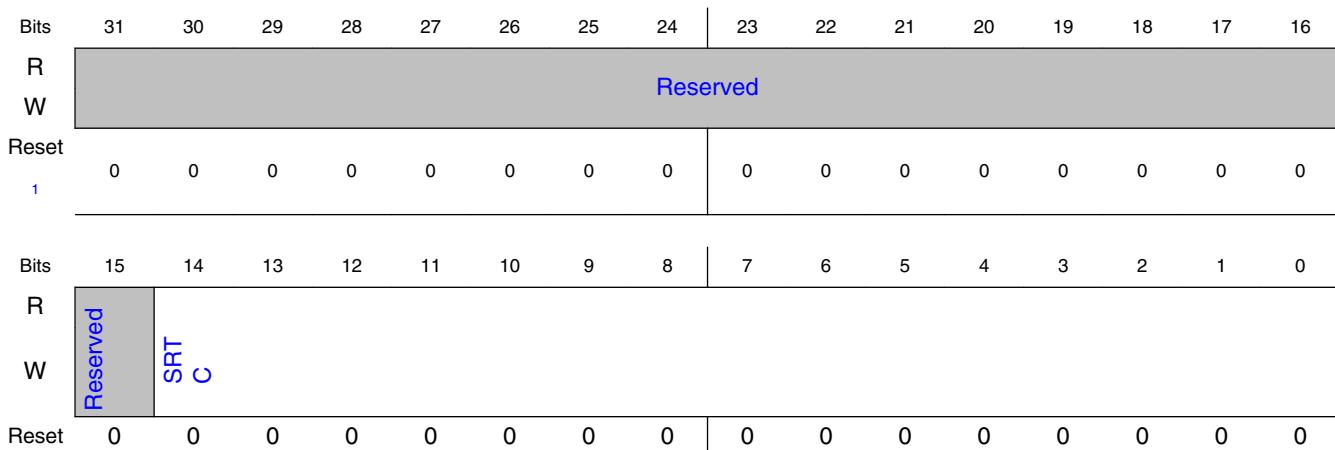
10.8.21.1 Offset

Register	Offset
LPSRTCMR	50h

10.8.21.2 Function

The SNVS_LP Secure Real Time Counter MSB register contains the 15 most-significant bits of the LP Secure Real Time Counter. This is a privileged write register.

10.8.21.3 Diagram



1. This register is reset at LP POR, but it is not reset by an LP software reset (i.e. writing a 1 to LP_SWR in HPCOMR).

10.8.21.4 Fields

Field	Function
31-15 —	Reserved.
14-0 SRTC	LP Secure Real Time Counter The most-significant 15 bits of the SRTC. This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

10.8.22 SNVS_LP Secure Real Time Counter LSB Register (LPSRTCLR)

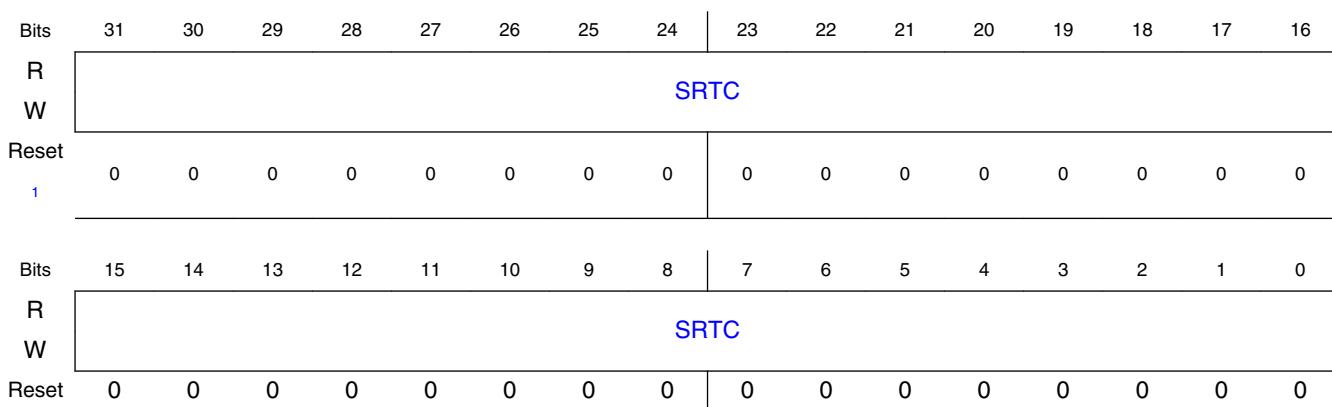
10.8.22.1 Offset

Register	Offset
LPSRTCLR	54h

10.8.22.2 Function

The SNVS_LP Secure Real Time Counter LSB register contains the 32 least-significant bits of the secure real time counter. This is a privileged write register.

10.8.22.3 Diagram



1. This register is reset at LP POR, but it is not reset by an LP software reset (i.e. writing a 1 to LP_SWR in HPCOMR).

10.8.22.4 Fields

Field	Function
31-0	LP Secure Real Time Counter least-significant 32 bits
SRTC	This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

10.8.23 SNVS_LP Time Alarm Register (LPTAR)

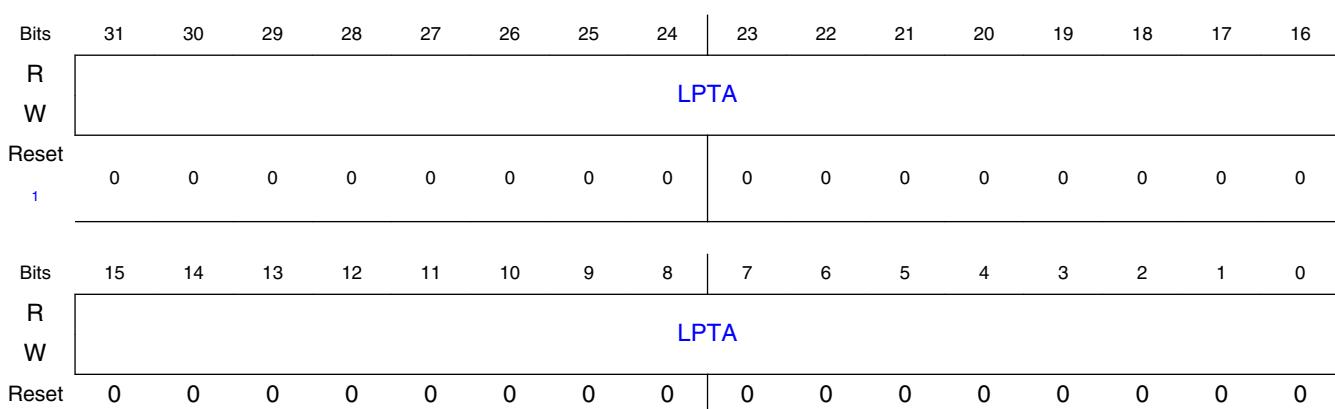
10.8.23.1 Offset

Register	Offset
LPTAR	58h

10.8.23.2 Function

The SNVS_LP Time Alarm register contains the 32-bit LP Time Alarm value. This is a privileged write register.

10.8.23.3 Diagram



1. This register is reset at LP POR, but it is not reset by an LP software reset (i.e. writing a 1 to LP_SWR in HPCOMR).

10.8.23.4 Fields

Field	Function
31-0	LP Time Alarm
LPTA	This register can be programmed only when the LP time alarm is disabled (LPTA_EN bit is not set).

10.8.24 SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR)

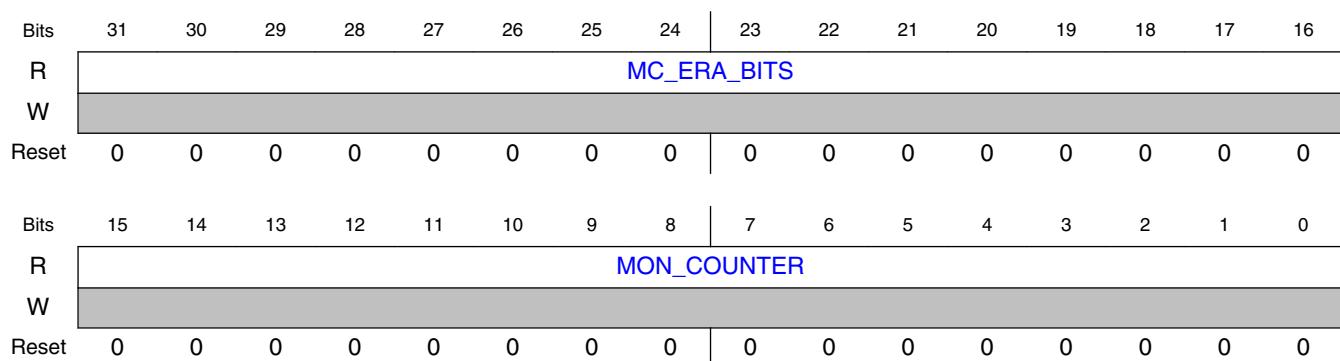
10.8.24.1 Offset

Register	Offset
LPSMCMR	5Ch

10.8.24.2 Function

The SNVS_LP Secure Monotonic Counter MSB Register contains the monotonic counter era bits and the most-significant 16 bits of the monotonic counter. The monotonic counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register. This is a non-privileged read-only register.

10.8.24.3 Diagram



10.8.24.4 Fields

Field	Function
31-16 MC_ERA_BITS	Monotonic Counter Era Bits These bits are inputs to the module and typically connect to fuses. When the Monotonic Counter is in use (i.e. enabled and valid and powered by an uninterrupted power source, e.g. a coin cell battery), and the boot software detects that the Monotonic Counter most-significant 16 Bits and Monotonic Counter LSB Register have been reset (MC_ENV=0), the boot software can take action to ensure that the value in the monotonic counter remains monotonic (i.e. never decreasing). The action is to blow an additional MC_ERA_BITS fuse. Since the MC_ERA_BITS field forms the most-significant field of the monotonic counter, blowing an additional fuse guarantees that the new monotonic counter value is higher than any previous value. Typically this would be necessary only when the coin cell battery is changed. Since the Monotonic Counter is reset on an LP Software Reset, an excessive number of MC_ERA_BITS fuses may be consumed if LP Software Reset is used repeatedly.
15-0 MON_COUNTE R	Monotonic Counter most-significant 16 Bits The MC is incremented by one when: <ul style="list-style-type: none"> • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set. This value can be reset only by LP software reset or LP POR.

10.8.25 SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR)

10.8.25.1 Offset

Register	Offset
LPSMCLR	60h

10.8.25.2 Function

The SNVS_LP Secure Monotonic Counter LSB Register contains the 32 least-significant bits of the monotonic counter. The MC is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register. This is a non-privileged read-only register.

10.8.25.3 Diagram

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

10.8.25.4 Fields

Field	Function
31-0 MON_COUNTE R	<p>Monotonic Counter bits</p> <p>The MC is incremented by one when:</p> <ul style="list-style-type: none"> A write transaction to the LPSMCMR or LPSMCLR Register is detected. The MC_ENV bit is set. MC_SL and MC_HL bits are not set. <p>This value can be reset only by LP software reset or LP POR.</p>

10.8.26 SNVS_LP Power Glitch Detector Register (LPPGDR)

10.8.26.1 Offset

Register	Offset
LPPGDR	64h

10.8.26.2 Function

The SNVS_LP Power Glitch Detector Register is a 32-bit read/write register that is used for storing the power glitch detector value, as described in [Power glitch detector \(PGD\)](#). This is a privileged write register.

10.8.26.3 Diagram

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

10.8.26.4 Fields

Field	Function
31-0	Power Glitch Detector Value
PGD	

10.8.27 SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias)

10.8.27.1 Offset

Register	Offset
LPGPR0_legacy_alias	68h

10.8.27.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in SNVS_LP, which can be used by any application for retaining data during an SoC power-down mode. This is a privileged read/write register. The full GPR register is accessed as 4 32-bit registers located in successive word addresses starting at offset 100h. For backward compatibility with earlier versions of SNVS, LPGPR0..LPGPR3 are aliased at

the earlier offset of 90h and LPGPR0 is aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

10.8.27.3 Diagram

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

10.8.27.4 Fields

Field	Function
31-0	General Purpose Register
GPR	When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

10.8.28 SNVS_LP Zeroizable Master Key Register (LPZMKR0 - LPZMKR7)

10.8.28.1 Offset

For a = 0 to 7:

Register	Offset
LPZMKRa	6Ch + (a × 4h)

10.8.28.2 Function

The SNVS_LP Zeroizable Master Key Registers contain the 256-bit zeroizable master key value. This is a privileged write register. These registers are programmable as follows:

- When ZMK write lock bit is set, they cannot be programmed.
- When ZMK_HWP is not set, they are in software programming mode and can be programmed only by software.
- When ZMK_HWP is set, they are in hardware programming mode and can be programmed only by hardware.

These registers cannot be read by software when the ZMK_HWP or ZMK read lock bit is set.

10.8.28.3 Diagram

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

10.8.28.4 Fields

Field	Function
31-0	Zeroizable Master Key
ZMK	Each of these registers contains 32 bits of the 256-bit ZMK value. The least-significant byte is at offset 6Ch.

10.8.29 SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_aalias - LPGPR3_alias)

10.8.29.1 Offset

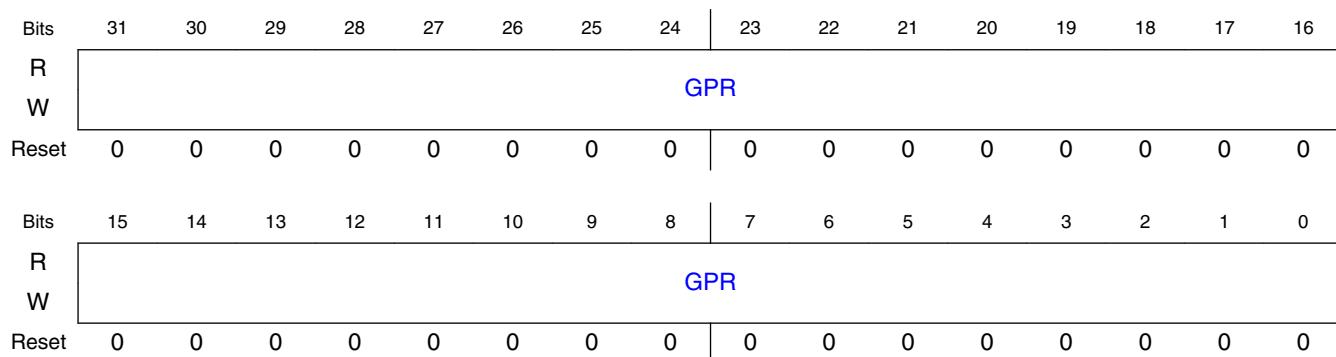
For $a = 0$ to 3:

Register	Offset
LPGPRa_alias	90h + ($a \times 4h$)

10.8.29.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in SNVS_LP, which can be used by any application for retaining data during an SoC power-down mode. This is a privileged read/write register. The full GPR register is accessed as 4 32-bit registers located in successive word addresses starting at offset 100h. For backward compatibility with earlier versions of SNVS, LPGPR0..LPGPR3 are aliased at the earlier offset of 90h and LPGPR0 is aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

10.8.29.3 Diagram



10.8.29.4 Fields

Field	Function
31-0	General Purpose Register
GPR	When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

10.8.30 SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0 - LPGPR3)

10.8.30.1 Offset

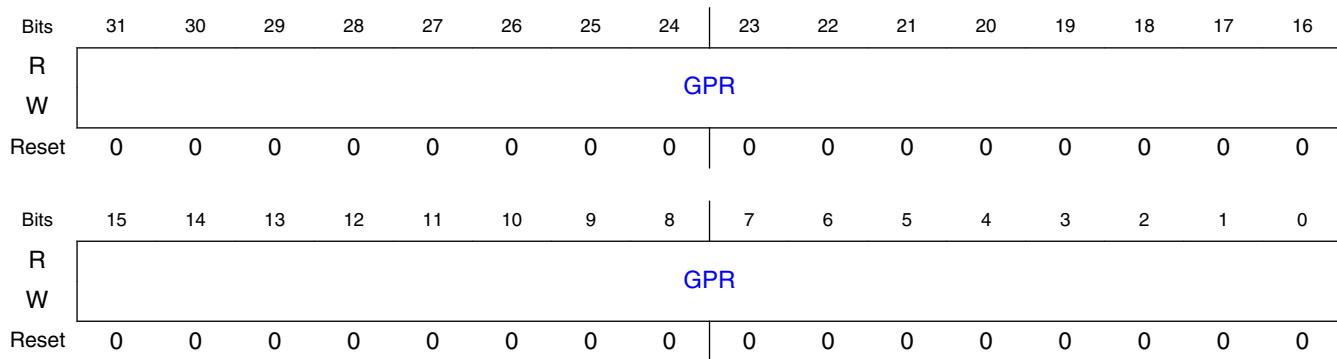
For $a = 0$ to 3 :

Register	Offset
LPGPRa	$100h + (a \times 4h)$

10.8.30.2 Function

The SNVS_LP General Purpose Register is a 128-bit read/write register located in SNVS_LP, which can be used by any application for retaining data during an SoC power-down mode. This is a privileged read/write register. The full GPR register is accessed as 4 32-bit registers located in successive word addresses starting at offset 100h. For backward compatibility with earlier versions of SNVS, LPGPR0..LPGPR3 are aliased at the earlier offset of 90h and LPGPR0 is aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

10.8.30.3 Diagram



10.8.30.4 Fields

Field	Function
31-0	General Purpose Register
GPR	When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

10.8.31 SNVS_HP Version ID Register 1 (HPVIDR1)

10.8.31.1 Offset

Register	Offset
HPVIDR1	BF8h

10.8.31.2 Function

The SNVS_HP Version ID Register 1 is a non-privileged read-only register that contains the current version of the SNVS. The version consists of a module ID, a major version number, and a minor version number.

10.8.31.3 Diagram

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

10.8.31.4 Fields

Field	Function
31-16 IP_ID	SNVS block ID
15-8 MAJOR_REV	SNVS block major version number
7-0 MINOR_REV	SNVS block minor version number

10.8.32 SNVS_HP Version ID Register 2 (HPVIDR2)

10.8.32.1 Offset

Register	Offset
HPVIDR2	BFCh

10.8.32.2 Function

The SNVS_HP Version ID Register 2 is a non-privileged read-only register that indicates the current version of the SNVS. Version ID register 2 consists of the following fields: integration options, ECO revision, and configuration options.

10.8.32.3 Diagram

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

10.8.32.4 Fields

Field	Function
31-24 IP_ERA	IP Era 00h - Era 1 or 2 03h - Era 3 04h - Era 4 05h - Era 5
23-16 INTG_OPT	SNVS Integration Options
15-8 ECO_REV	SNVS ECO Revision
7-0 CONFIG_OPT	SNVS Configuration Options

Chapter 11

System JTAG Controller (SJC)

11.1 Overview

The System JTAG Controller (SJC) provides debug and test control with the maximum security.

The test access port (TAP) is designed to support features compatible with the IEEE Standard 1149.1 v2001 (JTAG).

11.1.1 Features

The System JTAG Controller (SJC) provides the following capabilities:

- JTAG IEEE1149.1 mandatory instructions, see [EXTEST Instruction](#), [SAMPLE/ PRELOAD Instruction](#) , and [BYPASS Instruction](#) .
- JTAG IEEE1149.1 optional instructions, see [ID_CODE Instruction \(IDCODE\)](#) , and [HIGHZ Instruction](#).
- JTAG IEEE P1149.1 (standard JTAG) interface to off-chip test and development equipment including an SJC-only mode for true IEEE 1149.1 compliance, used primarily for board-level implementation of boundary scan.
- IEEE P1149.6 (JTAG) mandatory instructions, see [EXTEST_PULSE instruction](#) and [EXTEST_TRAIN instruction](#). These two instructions enable edge-detecting behavior on the signal path containing AC pins.
- Debug-related control and status, such as putting selected cores into reset and/or debug mode and the ability to monitor individual core status signals via JTAG.
- ExtraDebug logic (see [ENABLE_ExtraDebug Instruction](#)).
- Core compliant modes to support standalone core debuggers (see [Modes of Operation](#)).

11.1.2 Modes of Operation

The SJC modes are controlled through the MOD input port.

The MOD port (typically connected to pad of the same name) selects between two possible topologies of TAP connections, as seen at SoC level:

- Negating it (this should be the default state) selects CM7 DAP.
- Asserting it only selects the SJC TAP to be connected in the TDI-TDO chain.

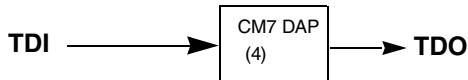
IEEE1149.1 standard features are enabled by configuring the SJC input pin: MOD. Refer to the following table for MOD settings details:

Table 11-1. SJC Modes

MOD	Name	Description
0	CM7 DAP	For common SW debug (High speed and production)
1	SJC only	IEEE 1149.1 JTAG compliant mode

The following figure shows the SJC mode selection flow. The numbers shown in parenthesis below each block name indicates the TAP's IR length.

MOD = 0



MOD = 1



(number in brackets lists IR length of given TAP)

Figure 11-1. SJC Mode Selection using MOD Pin Sampling

11.2 External Signals

The table found here describes the external signals of SJC.

Table 11-2. SJC External Signals

Signal	Description	Pad	Mode	Direction
JTAG_DE_B	SoC debug request/acknowledge pin. The DE_IN_B pin is used to propagate an external debug request event to the core(s). This functionality must be enabled first, by set of DE_to_ARM bit in SJC's DCR register. It is SoC implementation dependent, whether this pin can also be used to reflect the debug acknowledge event back from the cores (in the case where an Open-Drain scheme is used externally).	GPIO_EMC_01	ALT7	IO
JTAG_MOD	SJC mode selection. This pin is sampled at TRST reset to determine two possible modes for the TAP connection configuration.	GPIO_AD_B0_08	ALT0	I
JTAG_TCK	Test Clock (TCK). This is used to synchronize the test logic and includes an internal pull-up resistor	GPIO_AD_B0_07	ALT0	I
JTAG_TDI	Test Data Input (TDI). Serial test instruction and data are received through the test data input (TDI) pin. TDI is sampled on the rising edge of TCK and includes an internal pullup resistor	GPIO_AD_B0_09	ALT0	I
JTAG_TDO	Test Data Output (TDO). The serial output for test instructions and data. TDO is tri-stable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK	GPIO_AD_B0_10	ALT0	O
JTAG_TMS	Test Mode Select (TMS). This is used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK and includes an internal pullup resistor	GPIO_AD_B0_06	ALT0	I
JTAG_TRSTB	Test Reset (TRST). This is used to asynchronously initialize the test controller. The TRST pin has an internal pullup resistor	GPIO_AD_B0_11	ALT0	I

11.2.1 External Signal Overview

The SJC provides test and debug control with a minimum number of contacts.

The figure below shows SJC connections to external contacts and other chip blocks.

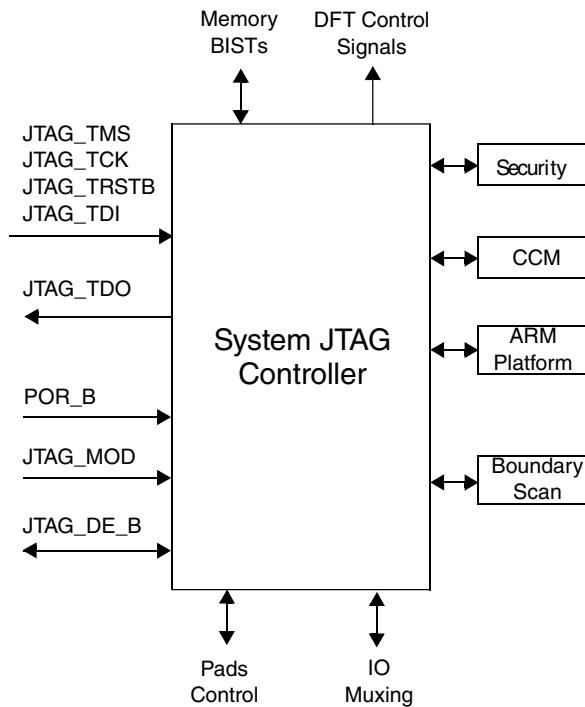
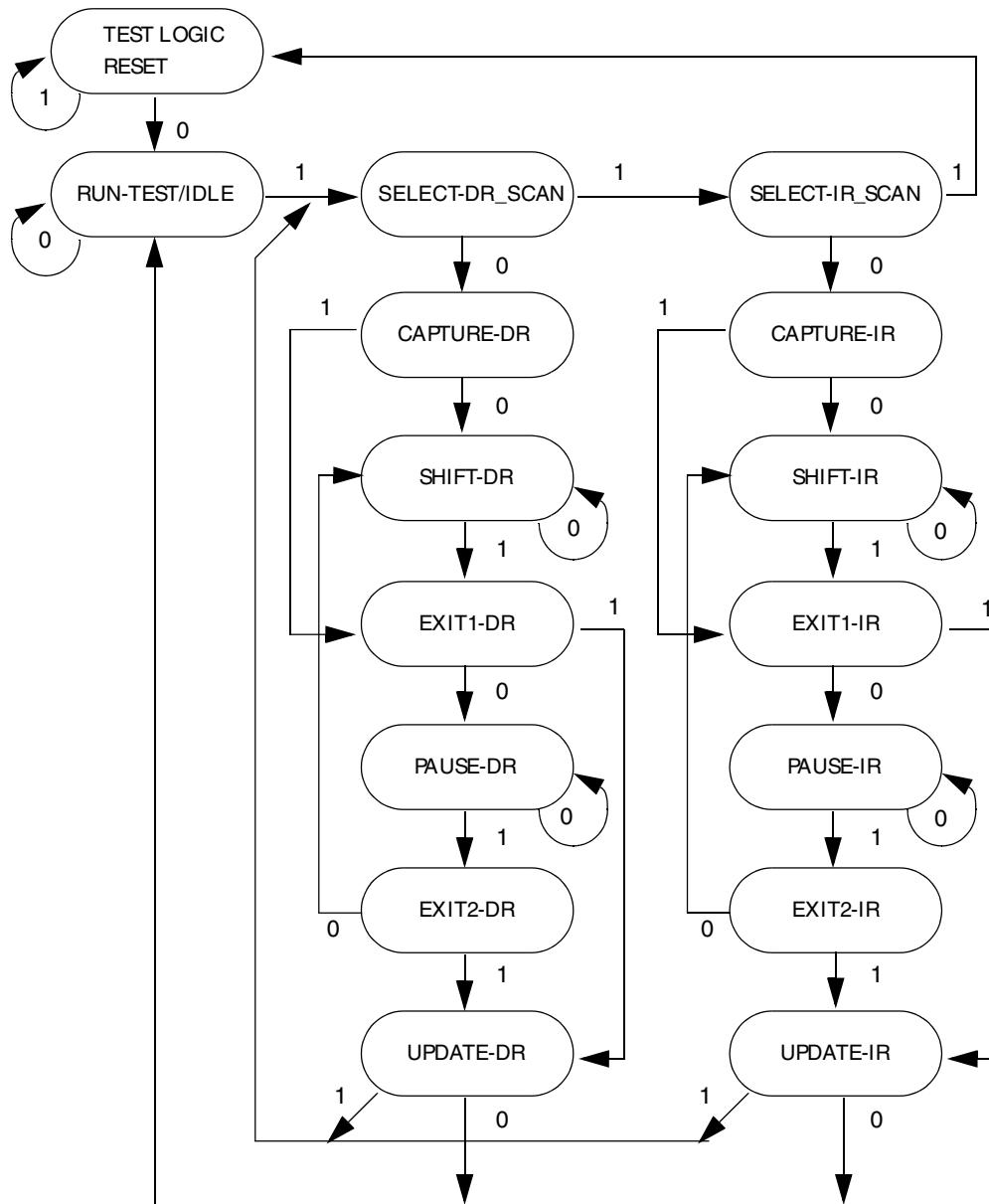


Figure 11-2. SJC Connections

11.2.2 TAP Controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, refer to the appropriate IEEE 1149.1 document.

The state machine is shown in the following figure.

**Figure 11-3. TAP Controller State Machine**

The change of the JTAG state machine occurs on the rising edge of TCK. TMS and TDI change on the falling edge of TCK. TDO also changes on the falling edge of TCK following entry into the Shift_DR or Shift_IR states (TDO_EN is the enable of the tristate buffer driving the TDO output).

The figure below shows the timings of the SJC signals.

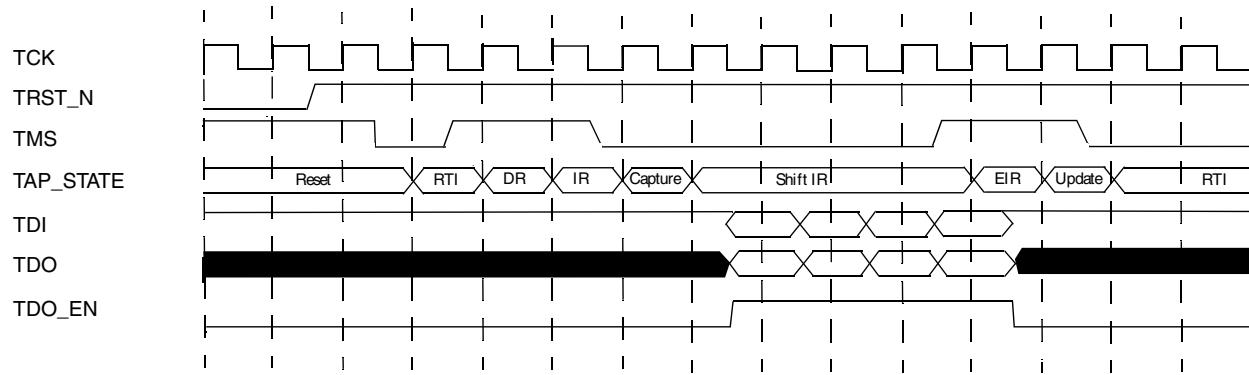


Figure 11-4. SJC Signals Timing Diagram

11.2.3 Accessing ExtraDebug Registers

Accessed through the Select-DR-Scan path, the ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bit data field (max length, see extra debug register description), a 5 bit address field and read/write bit.

The write actually takes place when the JTAG TAP controller enters the Update-DR state. On a read, the data field is ignored (the user should shift only 5 times to enter Read=1 and the address), the read takes place on the next path through DR at the Capture-DR state, the data is shifted-out during the Shift-DR state.

On the second path for a read access, simultaneous write access is not supported: command converter software shifts in zeros so the TAP decodes a write to the CSR (read-only register) which does not have any effect on the circuit.

The number of shift depends on the width of the accessed register as explained in the following diagrams.

First a write access (one path through Select-DR-Scan):

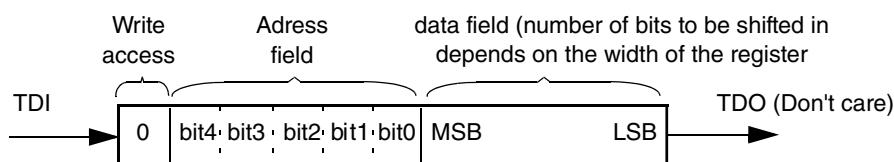
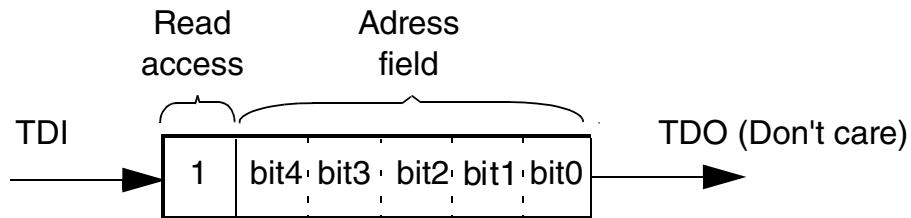


Figure 11-5. TDI/TDO on write access

Then a read access (requires two paths through Jtag DR Scan path):

First path



Second path

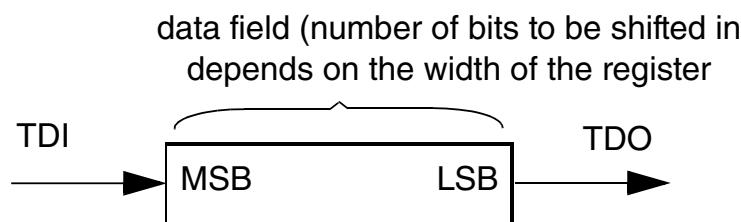


Figure 11-6. TDI/TDO on Read Access

For example, write value 0b1010_1100 to Debug Control Register (address = 0b00110).

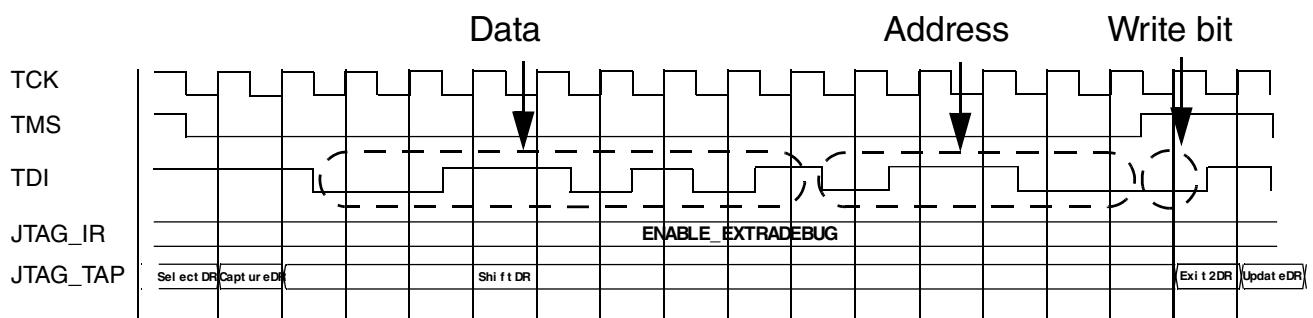


Figure 11-7. Example: Write Access to DCR

The SJC registers have different levels of security (Refer to [JTAG Security Modes](#)):

- Secured- accessible only in mode 2 (supposed correct response entered), mode 3 and mode 4.
- Unsecured- accessible in all modes

The level of security of each register is indicated in its name or description, in "Programmable Registers" section.

Boundary Scan Register (BSR)

A single DE_B pin is dedicated for debug request input/output in bidirectional open drain functionality (including an internal pull-up device).

Bit 6 in DCR register serves as mask bit, controlling the propagation of external debug request to ARM Platform.

The bit0 defines the propagation enable of IR debug request to recipient .

For security reasons, bits for output and input propagation control are at their negated values after reset. A user cannot put the core in debug mode through DE_B without any Jtag access.

The configuration after reset prevents propagation of debug requests / acknowledges to or from the core.

11.3 Boundary Scan Register (BSR)

The Boundary Scan Register (BSR) in the JTAG implementation contains bits for all device signal and clock pins and associated control signals.

All SoC bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register.

11.4 SoC JTAG Instruction Register (SJIR)

The SoC JTAG Instruction register is provided in the following table. The SoC JTAG Instruction register is 5 bits wide.

Table 11-3. SoC JTAG Instruction Register (SJIR)

Code					SJC IR
B4	B3	B2	B1	B0	
0	0	0	0	0	IDCODE
0	0	0	0	1	SAMPLE/PRELOAD
0	0	0	1	0	EXTTEST
0	0	0	1	1	HI-Z
0	0	1	0	0	ENABLE_ExtraDebug
0	0	1	0	1	ENTER_DEBUG (secured)
0	0	1	1	0	Reserved
0	0	1	1	1	Reserved
0	1	0	0	0	EXTTEST_PULSE

Table continues on the next page...

**Table 11-3. SoC JTAG Instruction Register (SJIR)
(continued)**

Code					SJC IR
B4	B3	B2	B1	B0	
0	1	0	0	1	EXTTEST_TRAIN
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Security Output challenge
0	1	1	0	1	Security Enter response
-					Reserved
1	1	1	1	1	BYPASS

The instruction register is reset to 0b00000 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.

During the capture-IR controller state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the standard; the most significant bits are loaded with the values 00, leading to a capture value of 0b000001.

11.4.1 ID_CODE Instruction (IDCODE)

Selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP.

The table below shows the ID register configuration.

Table 11-4. ID Configuration Register (IDCODE)

IDCODE				ID Configuration Register															
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16			
	Version Information[3:0]				Part Number (Bits 27-16)														
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r			
RESET	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x			
Note:																			
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0			
	Part Number (Bits 15-12)				Manufacturer Identity												1		
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r			
RESET	x	x	x	x	0	0	0	0	0	0	0	1	1	1	0	1			
Note:																			

Table 11-5. ID Configuration Register Description (IDCODE)

Field	Description
31-28 Version Information	IC/SoC Version information number. Initial value: '0000' This number is subject to changes, for new IC/SoC (System On A Chip) revision releases.
27-12 Part Number	Customer Part Number The 16-bit Part Number value is unique for every NXP SoC / IC. See System Debug chapter for exact register value for a specific SoC.
11-1 Manufacturer Identity	Manufacturer Identity NXP Manufacturer Identity code. Bits [11:1] - 00000001110
0	Tied to logic 1.

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design to determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Once the IDCODE instruction is decoded, it selects the ID register which is a 32 Bit data register. Because the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state shows whether such a register is included in the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by the IEEE 1149.1 standard.

11.4.2 SAMPLE/PRELOAD Instruction

Selects the boundary scan register and the system logic controls the I/O pins.

The SAMPLE/PRELOAD instruction provides two separate functions:

- First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.
- The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data appears on the outputs when entering the EXTEST instruction.

NOTE

Because there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

For more details on the function and use of SAMPLE/PRELOAD, refer to the appropriate IEEE 1149.1 document.

11.4.3 EXTEST Instruction

Selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins.

By using the TAP controller, the register is capable of:

- Scanning user-defined values into the output buffers,
- Capturing values presented to input pins
- Controlling the direction of bidirectional pins,
- Controlling the output drive of tri-statable output pins.

For more details on the function and use of EXTEST, refer to the appropriate IEEE 1149.1 document.

The EXTEST instruction also asserts internal reset for the cores (through CCM, refer to [Figure 1](#)) to force a predictable internal state while performing external boundary scan operations.

11.4.4 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (that is, high impedance). The instruction selects the bypass register.

In this mode, all internal pullup resistors on all the pins (except for the TMS, TDI, TCK, TRSTB pins) are disabled. This disabling functionality is not built into SJC, but should be implemented by some logic in the SOC/IO Pads.

For more details on the function and use of HIGHZ, refer to the IEEE 1149.1 document.

The HIGHZ instruction also asserts internal reset for the cores (through CCM, refer to [Figure 1](#)) to force a predictable internal state while performing external boundary scan operations.

11.4.5 BYPASS Instruction

Selects the single Bit bypass register and the system logic controls the I/O pins.

This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the boundary scan register. This instruction is used to enhance test efficiency when a component other than the SoC Core based device becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state.

Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero.

For more details on the function and use of BYPASS, refer to the appropriate IEEE 1149.1 document.

11.4.6 ENABLE_ExtraDebug Instruction

The TDI and TDO pins are connected directly to the ExtraDebug registers, the SJC TAP controller remaining connected to TDI and TMS.

The ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bits data field (maximum length, see [Accessing ExtraDebug Registers](#),), a 5 bits address field and read/write bit. On a register read, the data field does not need to be filled in. The particular ExtraDebug register connected between TDI and TDO at a given time is selected by the ExtraDebug controller depending on the ExtraDebug Address being currently decoded. All communication with the ExtraDebug controller is done through the Select-DR-Scan path of the JTAG TAP Controller.

11.4.7 ENTER_DEBUG instruction

The ENTER_DEBUG instruction is used to generate a debug request event to the ARM Core Platform .

The TDI and TDO are connected to the Instruction Register (IR). After the acknowledgment of the Debug Mode is received (can be checked by reading the Core Status Register part of the ExtraDebug logic), the user can perform system debug functions on the cores.

NOTE

The ENTER_DEBUG event issue to the cores, can be masked, by bits in DCR register.

NOTE

It is user's responsibility to shift-in another IR value (like IDCODE) before trying to bring the cores out of debug mode, as the debug request signals to the cores remains asserted as long as ENTER_DEBUG IR is in place.

NOTE

The user need to check that cores are in debug mode (watching debug acknowledge signal) before leaving ENTER_DEBUG instruction, otherwise debug request might not take affect.

11.4.8 EXTEST_PULSE instruction

The EXTEST_PULSE instruction implements test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

11.4.9 EXTEST_TRAIN instruction

The EXTEST_TRAIN instruction implements test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

11.5 Security

JTAG manipulation is one of the known hackers' ways of executing unauthorized program code, getting control over the OS and run code in privileged modes.

The SJC provides a debug access to several H/W blocks including the ARM processor and the system bus. This allows for program control and manipulation as well as visibility into system peripherals and memory. The ETM interface allows bus transactions to be traced. Together these tools provide the hacker all the access needed to completely comprise the system. Means must be provided to block any malicious JTAG access.

The SJC provides a way of regulating the JTAG access.

The following are the different JTAG security modes:

- **Mode #1: No Debug**-Maximum Security. All security sensitive JTAG features are permanently blocked.

- **Mode #2: Secure JTAG**-High security. JTAG use is regulated by secret key based authentication mechanism.
- **Mode #3: JTAG Enabled**-Low security. JTAG always enabled.

The JTAG security modes are configured using eFUSES which can be burned after packaging by applying electrical signals. The fuse burning is irreversible process, once a fuse is burned (e-fuse or laser fuse) it is impossible to change the fuse back to the unburned state.

11.5.1 JTAG Security Modes

JTAG can be in one of JTAG security modes which is selected by setting the SJC eFUSE configuration. The physical location of the fuses is not in the SJC.

11.5.1.1 Mode 1: No Debug - Maximum Security

No Debug JTAG security mode provides the highest security level.

In this mode, all JTAG features are disabled except for:

- ScanBoundary Scan
- MBIST, all modes except for debug modes which enable controlled memory contents output
- PLL BIST
- BIST monitor mode, allowing routing to external pins BIST pass/fail/invoke information
- PLL bypass- Bypass Arm or/and USB PLL.
- Visibility of the following status bits: power mode - normal, standby, stop, shutdown, and so on

These features do not reduce the security level of the product, and they allows to perform important tests and board connectivity checks.

11.5.1.2 Mode 2: Secure JTAG - High Security

The Secure JTAG mode limits the JTAG access by using challenge/response based authentication mechanism. Any access to JTAG port is being checked. Only authorized debug devices (that is, devices having the right response) can access the JTAG, unauthorized JTAG access attempts are denied.

The intent of this mode is to allow return field testing. When a secured JTAG device is being returned for debugging, this mode allows authorized re-activation of the JTAG.

11.5.1.2.1 Challenge/Response Mechanism in System JTAG Mode

When SJC is in Sysytem JTAG mode the authentication process is as follows:

1. Shift Output Challenge instruction to IR.
2. Passing through Capture-DR state of the SJC and by performing Shift-DR operations Challenge code can be accessed from TDO.
3. Shift Enter Response instruction to IR. By performing Shift-DR, operations enter Response code value through TDI. As Update-DR state is entered, Response code is compared with the correct one.

In Fixed challenge-response pair mode, each part has its individual challenge - response pair which is determined at manufacturing time, and does not change later on. The SJC compares the user's response to the expected response.

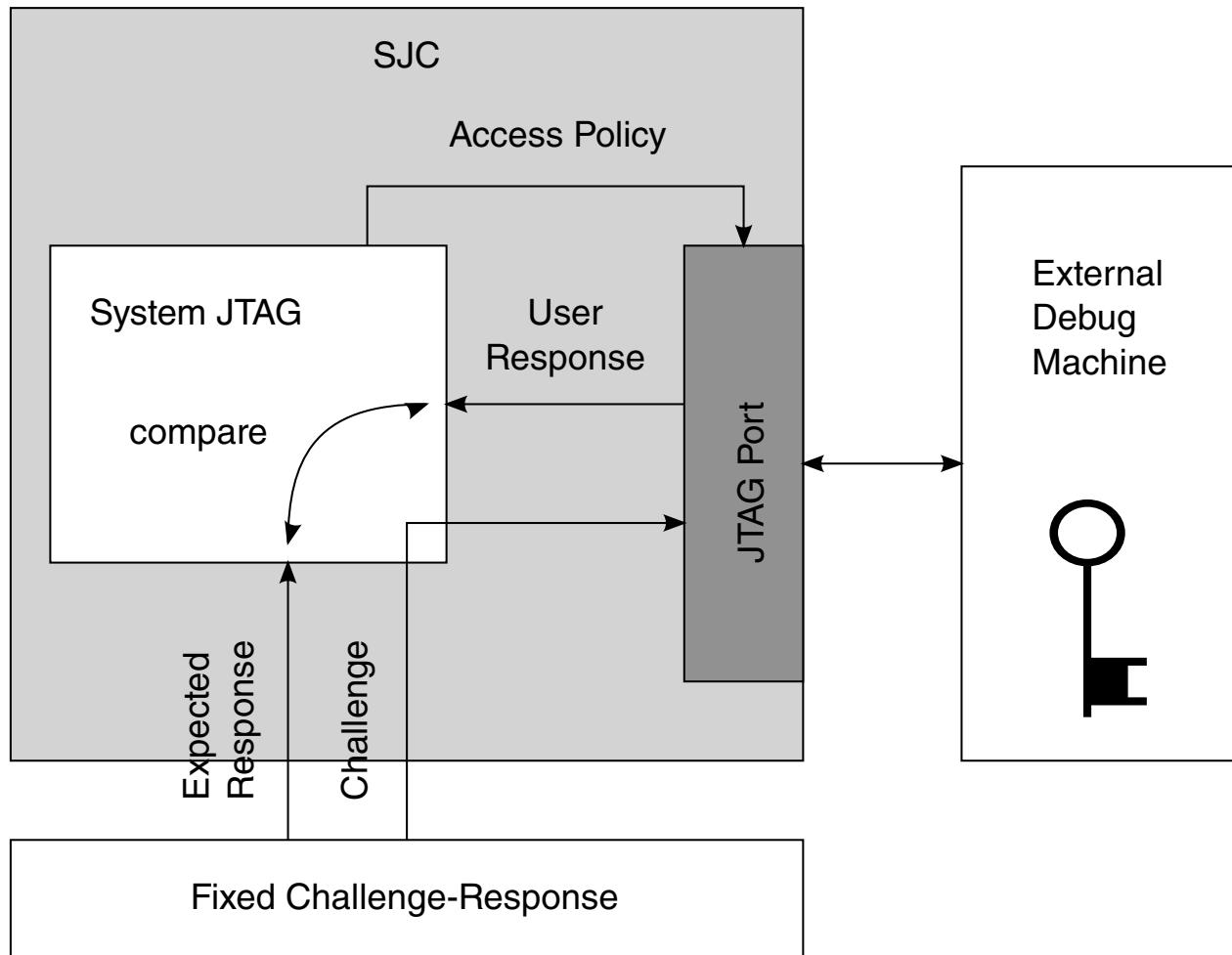


Figure 11-8. Mode #2 - Secure JTAG with Fixed Challenge-response Pair

11.5.1.3 Mode 3: JTAG Enabled - Low Security

In the JTAG Enabled JTAG security mode, all JTAG features are enabled.

11.5.2 Software Enabled JTAG

To increase the flexibility of the SJC, an option to enable the JTAG via software is added and is available only in Secure JTAG mode. By writing '1' to HAB_JDE (HAB JTAG DEBUG ENABLE) bit in the e-fuse controller module, the JTAG is opened, regardless of its security mode. It is the responsibility of software to assert or negate this bit.

Additionally, a corresponding lock bit is available (in the e-fuse control module) to ensure that only trusted software is able to set the JDE bit. When the LOCK bit is set, no future change of JDE is possible, until the next POR (power-on-reset) cycle.

The platform initialization software should set the LOCK bit for JDE bit before transferring control to the application code.

The S/W JTAG enable allows JTAG enabling without activating the challenge-Response mechanism (which requires JTAG access tool enhancement or special H/W). The JTAG S/W enable does not allow debug in case of boot or memory fault as it requires reset before entering debug.

This feature can be permanently blocked by burning the dedicated e-fuse.

NOTE

The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is strongly recommended to burn the JTAG_HEO e-fuse which disables this feature.

11.5.3 Kill Trace

The kill trace signal disables any output of the ETM block. The ETM can be accessed either via JTAG port and/or by direct software code. Blocking the JTAG port also yields assertion of the kill trace signal. This resulted in blocking of trace port. The intention of this action is to block any attempt to break into the system via software manipulation of the debug modules. The kill trace, when active, prevents trace output even in case where it can be activated via chip pin.

The kill trace feature needs to be activated by burning a dedicated e-fuse. If the fuse is left intact, kill trace is never activated as seen in [Figure 11-9](#).

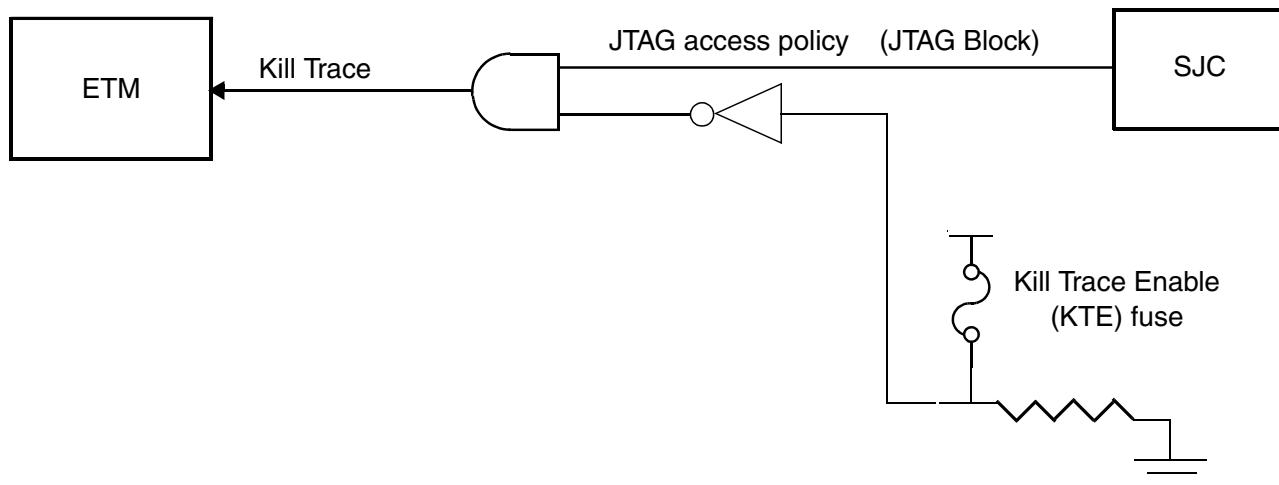


Figure 11-9. Kill Trace eFUSE

The kill trace is asserted when "kill trace enable" fuse is burned and "ipt_secur_block" signal in SJC is asserted, which happens when at least one of the following is true:

- Mode #2 (Secure JTAG) and no code has been entered
- Mode #2 (Secure JTAG) with burned Bypass and Re-enable fuses
- Mode #2 (Secure JTAG) with incorrect response entered
- Mode #1 (No debug)
- TRST_B signal is active
- POR has not ever been asserted

11.5.4 SJC Disable Fuse

In addition to the different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by eFUSE configuration. This creates additional JTAG mode that is, JTAG Disabled with highest level of JTAG protection. In this mode all JTAG features are disabled.

Specifically, the following debug features are disabled in addition to the features that were already disabled in No Debug JTAG mode:

- Memory BIST
- Boundary scan register (SJC_BSR)
- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

11.6 Functional Description

This section provides a complete functional description of the block.

11.6.1 Static Core Debug

The SJC JTAG TAP controller is fully compatible with the IEEE 1149.1a-2001 Standard Test Access Port and Boundary Scan Architecture specifications.

The ARM platform has an integrated JTAG interface and a TAP controller to manage its own ICE. Also it can access an embedded trace ETM interface, see ARM core and ETM Technical reference guide for more information.

11.6.2 Reset Mechanism

The following figure shows the SJC reset logic

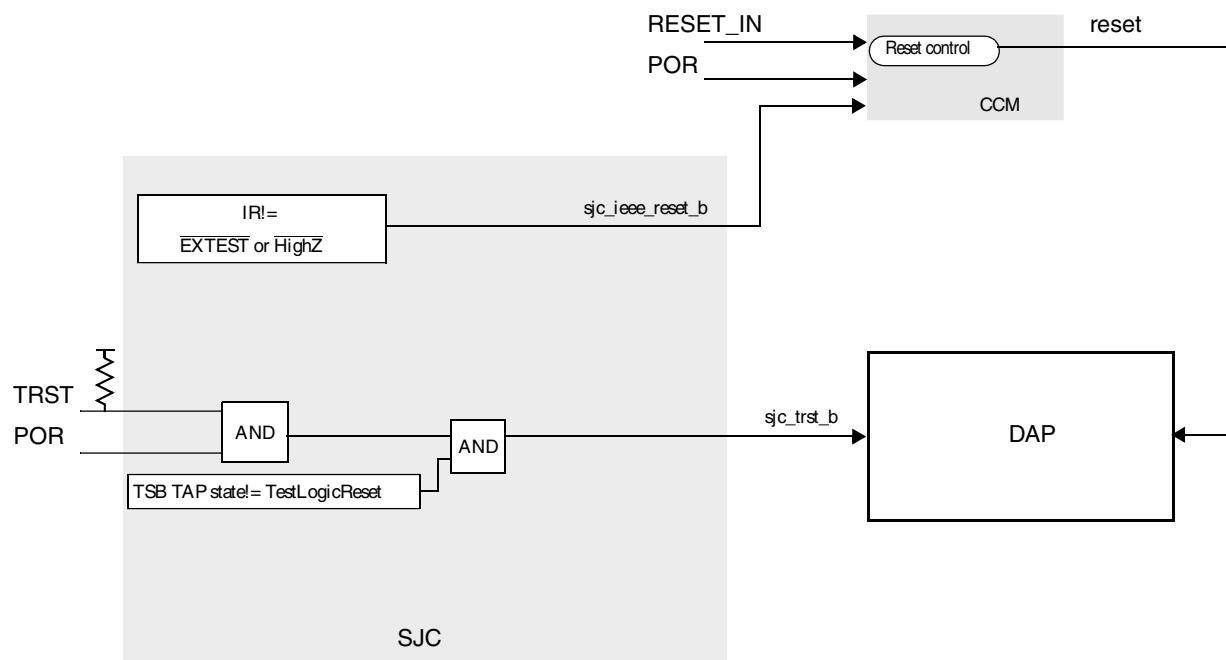


Figure 11-10. SJC Reset Logic

NOTE

- Asserting TRSTB in any scan mode resets the TCR loosing the testmode configuration and selects default TAP.
- SJC generates an IEEE reset signal to the CCM when in one of the IEEE modes HIGHZ or EXTEST. This signal generates a system reset to the cores until exit from one of these modes.

11.7 Initialization/Application Information

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the SJC output drivers are enabled into actively driven networks.

There are two constraints related to the JTAG interface:

- Ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the Test-Logic-Reset controller state. During power-up, SJC's internal TRSTB is asserted as IC's POR_B is asserted which forces the TAP controller into this state. After that, if TMS either remains unconnected or is connected to VCC, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.
- DE_B is an IO pin with pullup and care must be taken of the direction when driving this signal.

11.8 SJC Memory Map/Register Definition

In addition to the standard accessible JTAG registers (per IEEE1149.1 standard) listed in [SoC JTAG Instruction Register \(SJIR\)](#), the chip contains the following registers accessed using the ExtraDebug mechanism, controlled via "ENABLE_ExtraDebug" IR instruction.

NOTE

SJC registers are only accessible by JTAG interface. They are not memory mapped to processor address space, so the absolute addresses provided by default in the SJC memory map are not valid.

This section assumes the JTAG controller is accessed in standalone mode .

See "System Debug" chapter for more details about the general purpose register descriptions that are unique to this chip.

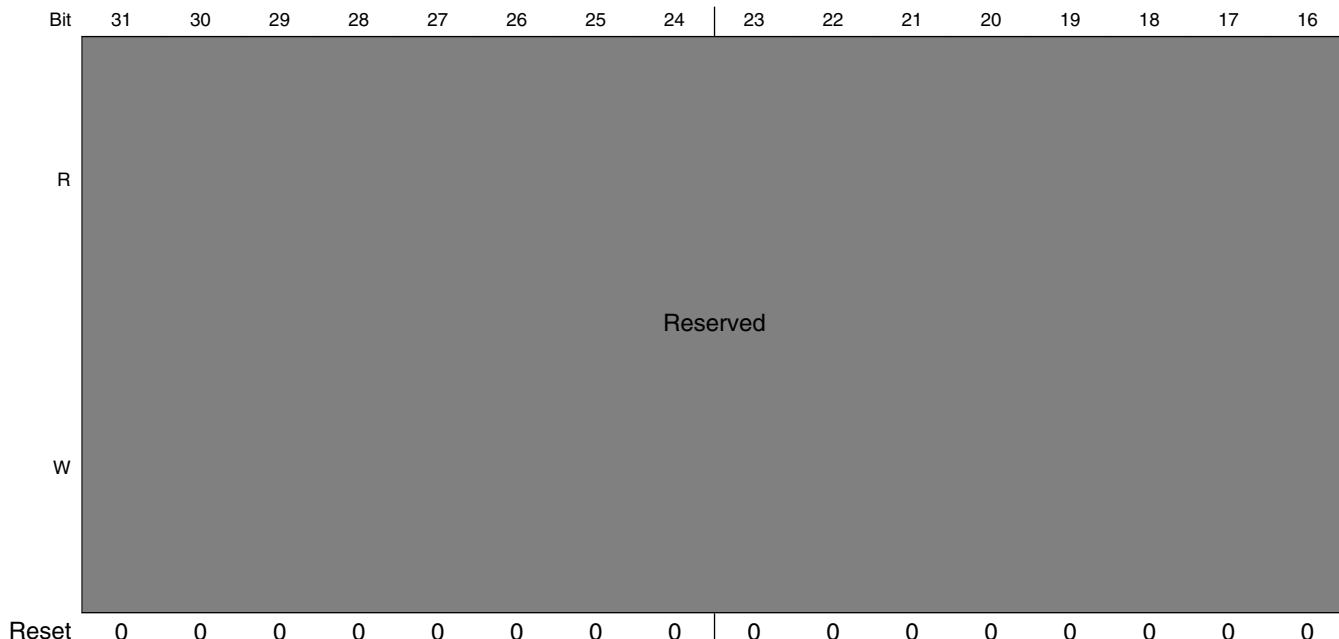
SJC memory map

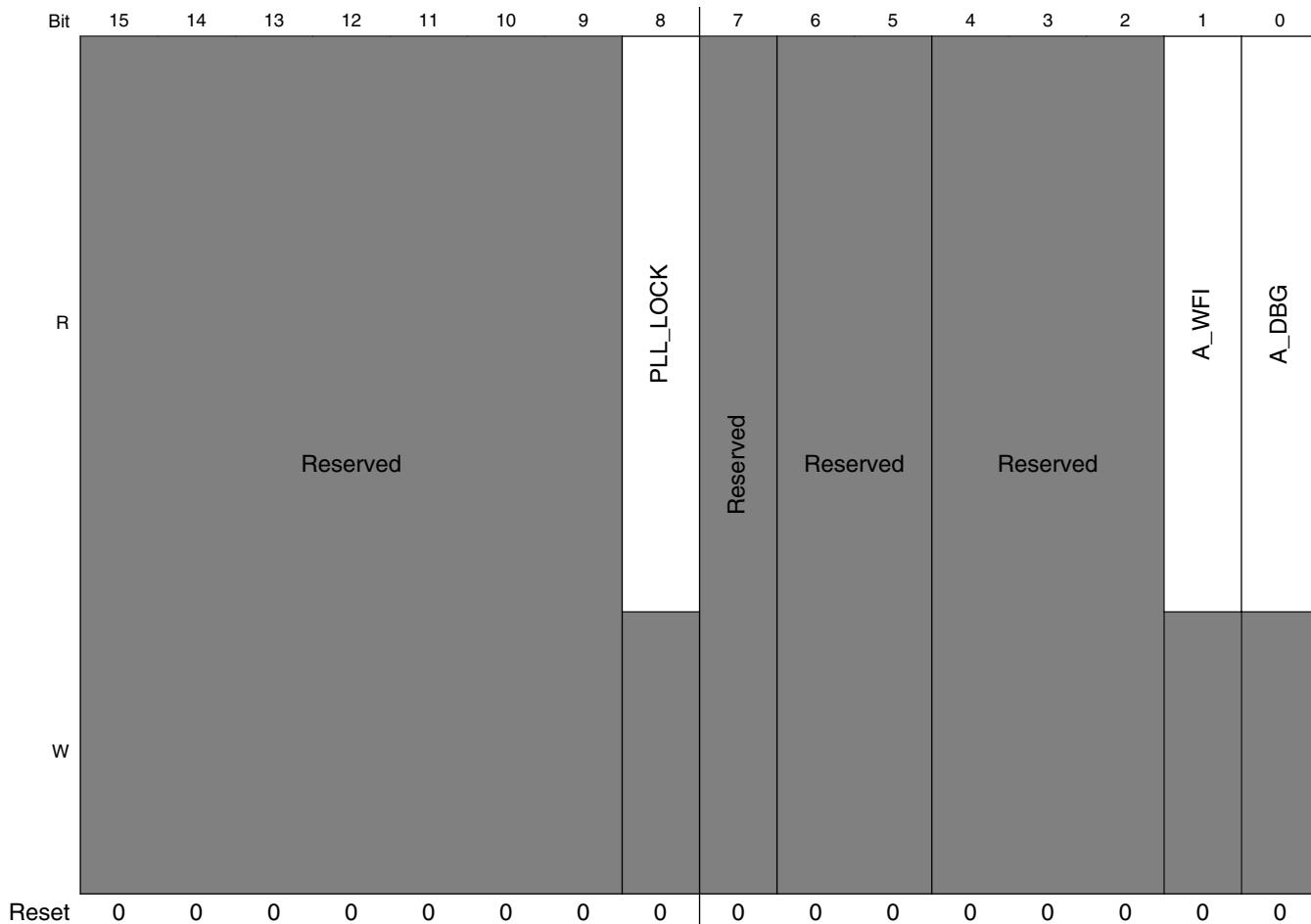
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
400E_4000	General Purpose Unsecured Status Register 1 (SJC_GPUSR1)	32	R	0000_0000h	11.8.1/516
400E_4001	General Purpose Unsecured Status Register 2 (SJC_GPUSR2)	32	R	0000_0000h	11.8.2/518
400E_4002	General Purpose Unsecured Status Register 3 (SJC_GPUSR3)	32	R	0000_0000h	11.8.3/519
400E_4003	General Purpose Secured Status Register (SJC_GPSSR)	32	R	0000_0000h	11.8.4/519
400E_4004	Debug Control Register (SJC_DCR)	32	R/W	1F9E_0000h	11.8.5/520
400E_4005	Security Status Register (SJC_SSR)	32	R	See section	11.8.6/522
400E_4007	General Purpose Clocks Control Register (SJC_GPCCR)	32	R/W	0000_0000h	11.8.7/525

11.8.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1)

The General Purpose Unsecured Status Register 1 is a read only register used to check the status of the different Cores and of the PLL. The rest of its bits are for general purpose use.

Address: 400E_4000h base + 0h offset = 400E_4000h



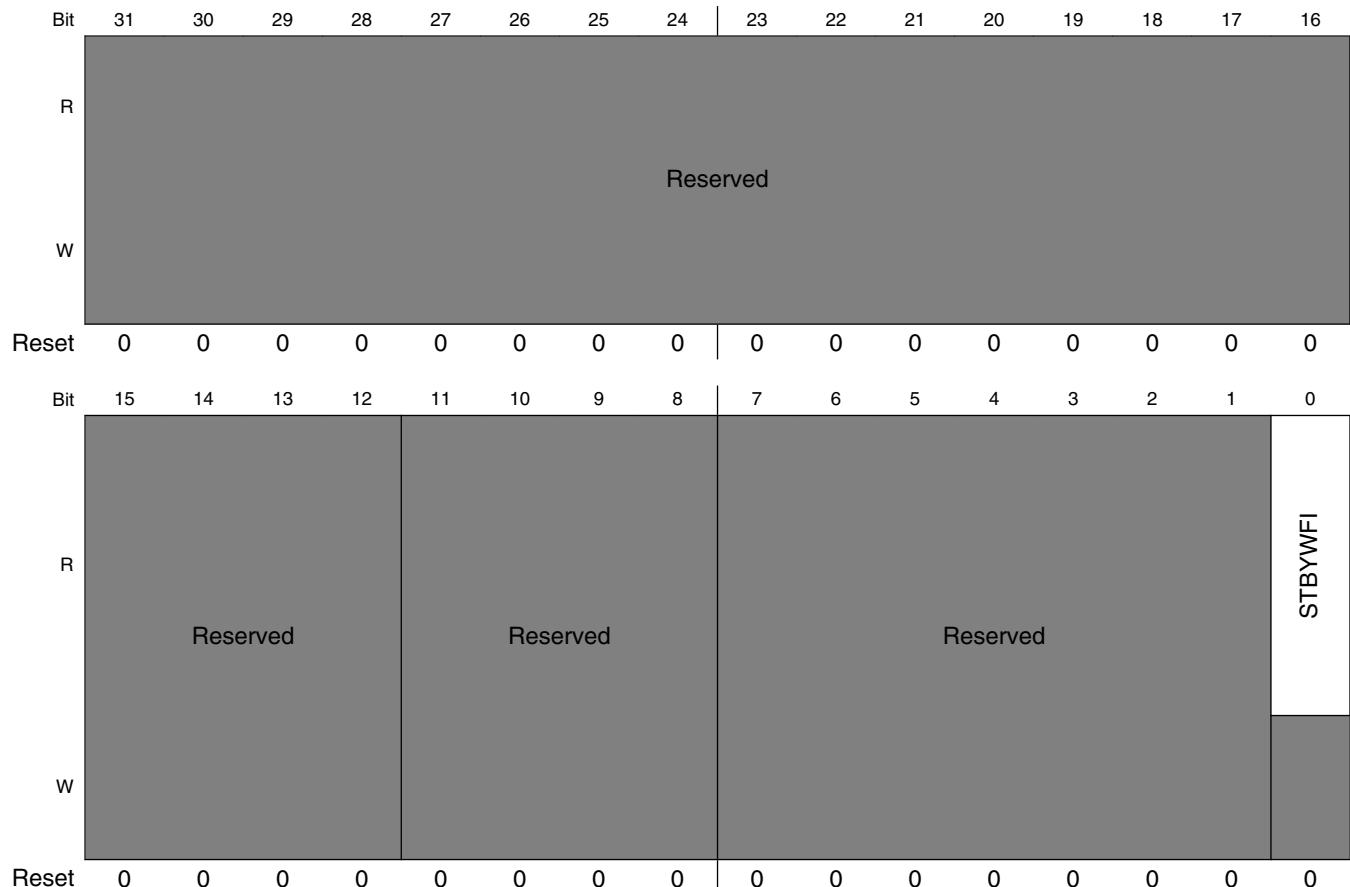


SJC_GPUSR1 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved.
8 PLL_LOCK	PLL_LOCK A Combined PLL-Lock flag indicator, for all the PLL's.
7 -	This field is reserved. Reserved
6–5 -	This field is reserved. Reserved.
4–2 -	This field is reserved. Reserved.
1 A_WFI	ARM core wait-for interrupt bit Bit 1 is the ARM core standbywfi (stand by wait-for interrupt). When this bit is HIGH, ARM core is in wait for interrupt mode.
0 A_DBG	ARM core debug status bit Bit 0 is the ARM core DBGACK (debug acknowledge) DBGACK can be overwritten in the ARM core DCR to force a particular DBGACK value. Consequently interpretation of the DBGACK value is highly dependent on the debug sequence. When this bit is HIGH, ARM core is in debug.

11.8.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2)

Address: 400E_4000h base + 1h offset = 400E_4001h



SJC_GPUSR2 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 -	This field is reserved. Reserved
7–1 -	This field is reserved. Reserved
0 STBYWFI	STBYWFI These bits provide status of "Standby Wait-For-Interrupt" state of ARM core.

11.8.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3)

Address: 400E_4000h base + 2h offset = 400E_4002h

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

SJC_GPUSR3 field descriptions

Field	Description
31–30	This field is reserved.
-	Reserved
2 SYS_WAIT	System In wait Indication on System in wait mode (from CCM).
1 IPG_STOP	IPG_STOP CCM's "ipg_stop" signal indication
0 IPG_WAIT	IPG_WAIT CCM's "ipg_wait" signal indication

11.8.4 General Purpose Secured Status Register (SJC_GPSSR)

The General Purpose Secured Status Register is a read-only register used to check the status of the different critical information in the SoC. This register cannot be accessed in secure modes.

Address: 400E_4000h base + 3h offset = 400E_4003h

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

SJC_GPSSR field descriptions

Field	Description
GPSSR	General Purpose Secured Status Register Register is used for testing and debug.

11.8.5 Debug Control Register (SJC_DCR)

This register is used to control propagation of debug request from DE_B pad to the cores and debug signals from internal logic to the DE_B pad.

Address: 400E_4000h base + 4h offset = 400E_4004h

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

SJC_DCR field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 DIRECT_ARM_REQ_EN	Pass Debug Enable event from DE_B pin to ARM platform debug request signal(s). This bit controls the propagation of debug request DE_B to the Arm platform. 0 Disable propagation of system debug to (DE_B pin) to Arm platform. 1 Enable propagation of system debug to (DE_B pin) to Arm platform.

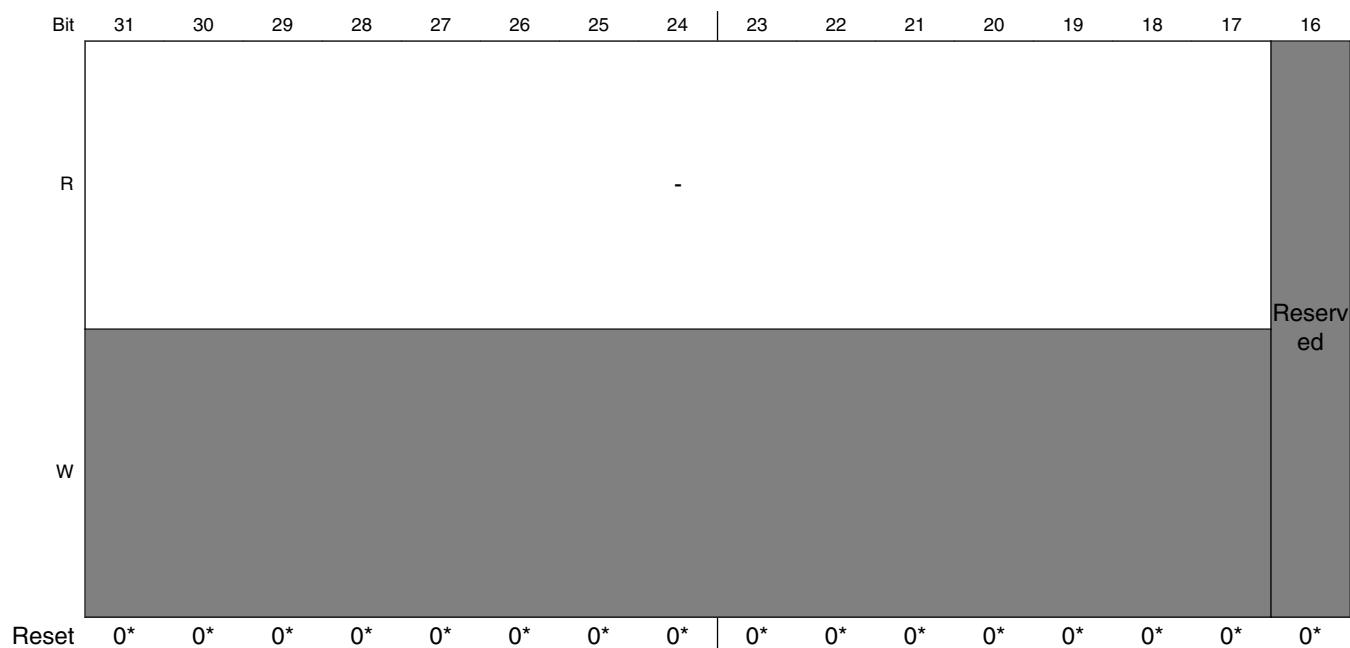
Table continues on the next page...

SJC_DCR field descriptions (continued)

Field	Description
5 -	This field is reserved. Reserved
4 -	This field is reserved. Reserved
3 DEBUG_OBS	Debug observability This bit controls the propagation of the "system debug" input to SJC (driven by the ECT logic), to the DE_B pad. (This logic can be used to pass debug acknowledge event from ECT out to the PAD, for example). The SJC's "system_debug" input is tied to logic HIGH value, therefore, set of "debug_obs" bit, will result in unconditional assertion of DE_B pad. 0 Disable propagation of system debug to DE_B pin 1 Unconditional assertion of pad DE_B
2 -	This field is reserved. Reserved
1 -	This field is reserved. Reserved
0 DE_TO_ARM	ARM platform debug request input propagation This bit controls the propagation of debug request to ARM platform ("dbgreq"), when the JTAG state machine is put in "ENTER_DEBUG" IR instruction. 0 Disable propagation of debug request to ARM platform 1 Enable propagation of debug request to ARM platform

11.8.6 Security Status Register (SJC_SSR)

Address: 400E_4000h base + 5h offset = 400E_4005h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	BOOTIND	Reserved	RSSTAT	SJM	FT	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	SWE	SWF	KTA	KTF
W																
Reset	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The SJM reset value, reflects the JTAG security state, as defined by status of JTAG_SMODE[1:0] fuses. See the [SJM](#) bitfield description for details on valid values.

SJC_SSR field descriptions

Field	Description
31–17 -	Reserved.
16–15 -	This field is reserved. Reserved
14 BOOTIND	Boot Indication Inverted Internal Boot indication, i.e inverse of SRC: "src_int_boot" signal
13 -	This field is reserved. Reserved
12–11 RSSTAT	Response status Response status bits 00 Response wasn't entered 01 Response was entered but not verified

Table continues on the next page...

SJC_SSR field descriptions (continued)

Field	Description
	10 Response was entered and is incorrect 11 Response is correct
10–9 SJM	SJC Secure mode Secure JTAG mode, as set by external fuses. 00 No debug (#1) 01 Secure JTAG (#2) 10 Reserved 11 JTAG enabled (#3)
8 FT	Fuse type Fuse type bit - e-fuse or laser fuse 0 E-fuse technology 1 Laser fuse technology
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5 -	This field is reserved. Reserved
4 -	This field is reserved. Reserved
3 SWE	SW enable SW JTAG enable status 1 enabled 0 disabled
2 SWF	Software JTAG enable fuse Status of the no SW disable JTAG fuse 0 (intact) - SW enable possible 1 (intact) - no SW enable possible
1 KTA	Kill Trace is active 1 active 0 not active
0 KTF	Kill Trace Enable fuse value 0 (intact) - kill trace is never active 1 (burned) - kill trace functionality enabled

11.8.7 General Purpose Clocks Control Register (SJC_GPCCR)

Address: 400E_4000h base + 7h offset = 400E_4007h

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

SJC_GPCCR field descriptions

Field	Description
31 SYS_RESET_REQ	Set to force a system reset. The system remains held in reset until this bit is cleared.
30–2 -	Reserved
-	Reserved

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm, Cortex, and TrustZone are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. CoreLink and CoreSight are trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2018 NXP B.V.

Document Number IMXRT1050SRM
Revision 1, 04/2018

arm

NXP