Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

FCR4 Cluster Series Hardware Manual

Doc No. 002-09388 Rev. *C

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Copyrights

© Cypress Semiconductor Corporation, 2011-2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Contents

1. Overview **29**

1.1 Overview 32
1.1.1 Features of the FCR4 Cluster series 32
1.2 General notes on using this document 35
1.3 Legend 36
1.3.1 Global description of register table 36
1.4 Abbreviations 37
1.5 Feature list 41
1.6 Memory map 48
1.7 Register access 55
1.7.1 Combined registers 55
1.7.2 Single bit access registers 56

2. System Controller **59**

2.1 Outline of System Controller 59
2.2 System Controller registers 61
2.2.1 Registers of System Controller 61
2.2.2 Memory layout of System Controller registers 64
2.2.3 Protection register 70
2.2.3.1 Protection Key Register (SYSC_PROTKEYR) 70
2.2.4 RUN profile register set 71
2.2.4.1 RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR) 71
2.2.4.2 RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER) 73
2.2.4.3 RUN Profile Clock Select Register (SYSC_RUNCKSELR) 75
2.2.4.4 RUN Profile Clock Enable Register (SYSC_RUNCKER) 78
2.2.4.5 RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0) 82
2.2.4.6 RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1) 84
2.2.4.7 RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2) 86
2.2.4.8 RUN Profile PLL Control Register (SYSC_RUNPLLCNTR) 88
2.2.4.9 RUN Profile SSCG Control Register 0 (SYSC_RUNSSCGCNTR0) 90
2.2.4.10 RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1) 92
2.2.4.11 RUN Profile Graphics PLL Control Register 0 (SYSC_RUNGFXCNTR0) 94
2.2.4.12 RUN Profile Graphics PLL Control Register 1 (SYSC_RUNGFXCNTR1) 96
2.2.4.13 RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR) 98
2.2.4.14 RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR) 101
2.2.4.15 Trigger RUN Control Register (SYSC_TRGRUNCNTR) 103
2.2.5 PSS profile register set 104

2.2.5.1 PSS Profile Power Domain Configuration Register (SYSC_PSSPDCFGR) 105
2.2.5.2 PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER) 106
2.2.5.3 PSS Profile Clock Select Register (SYSC_PSSCKSELR) 108
2.2.5.4 PSS Profile Clock Enable Register (SYSC_PSSCKER) 110
2.2.5.5 PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0) 115
2.2.5.6 PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1) 117
2.2.5.7 PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2) 119
2.2.5.8 PSS Profile PLL Control Register (SYSC_PSSPLLCNTR) 121
2.2.5.9 PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0) 123
2.2.5.10 PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1) 125
2.2.5.11 PSS Profile Graphics PLL Control Register 0 (SYSC_PSSGFXCNTR0) 127
2.2.5.12 PSS Profile Graphics PLL Control Register 1 (SYSC_PSSGFXCNTR1) 129
2.2.5.13 PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR) 131
2.2.5.14 PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR) 134
2.2.5.15 PSS Enable Register (SYSC_PSSENR) 136
2.2.6 APPLIED profile register set 137
2.2.6.1 APPLIED Profile Power Domain Configuration Register (SYSC_APPPDCFGR) 138
2.2.6.2 APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER) 139
2.2.6.3 APPLIED Profile Clock Select Register (SYSC_APPCKSELR) 141
2.2.6.4 APPLIED Profile Clock Enable Register (SYSC_APPCKER) 143
2.2.6.5 APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0) 146
2.2.6.6 APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1)7 148
2.2.6.7 APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2) 150
2.2.6.8 APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR) 152
2.2.6.9 APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0) 154
2.2.6.10 APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1) 156
2.2.6.11 APPLIED Profile Graphics PLL Control Register 0 (SYSC_APPGFXCNTR0) 158
2.2.6.12 APPLIED Profile Graphics PLL Control Register 1 (SYSC_APPGFXCNTR1) 160
2.2.6.13 APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR) 162
2.2.6.14 APPLIED Profile Clock Supervisor Configuration Register (SYSC_APPCSVCFGR) 165
2.2.7 Profile status register set 166
2.2.7.1 Power Domain Status Register (SYSC_PDSTSR) 166
2.2.7.2 Clock Source Enable Status Register (SYSC_CKSRESTSR) 167
2.2.7.3 Clock Select Status Register (SYSC_CKSELSTSR) 170
2.2.7.4 Clock Enable Status Register (SYSC_CKESTSR) 172
2.2.7.5 Clock Divider Register 0 (SYSC_CKDIVSTSR0) 175
2.2.7.6 Clock Divider Register 1 (SYSC_CKDIVSTSR1) 177
2.2.7.7 Clock Divider Register 2 (SYSC_CKDIVSTSR2) 179
2.2.7.8 PLL Status Register (SYSC_PLLSTSR) 181
2.2.7.9 SSCG PLL Status Register 0 (SYSC_SSCGSTSR0) 183

2.2.7.10 SSCG PLL Status Register 1 (SYSC_SSCGSTSR1) 185
2.2.7.11 Graphics PLL Status Register 0 (SYSC_GFXSTSR0) 187
2.2.7.12 Graphics PLL Status Register (SYSC_GFXSTSR1) 189
2.2.7.13 LVD Configuration Status Register (SYSC_LVDCFGSTSR) 191
2.2.7.14 CSV Configuration Status Registe (SYSC_CSVCFGSTSR) 194
2.2.8 System registers 195
2.2.8.1 System ID Register (SYSC_SYSIDR) 195
2.2.8.2 System Status Register (SYSC_SYSSTSR) 196
2.2.8.3 System Status Interrupt Enable Register (SYSC_SYSINTER) 199
2.2.8.4 System Status Interrupt Clear Register (SYSC_SYSICLR) 200
2.2.8.5 System Error Register (SYSC_SYSERRR) 201
2.2.8.6 System Error Interrupt Clear Register (SYSC_SYSERRICLR) 204
2.2.9 Clock Supervisor Configuration registers 207
2.2.9.1 Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR) 207
2.2.9.2 Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR) 209
2.2.9.3 Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR) 211
2.2.9.4 Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR) 213
2.2.9.5 Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR) 215
2.2.9.6 Clock Supervisor Test Register (SYSC_CSVTESTR) 217
2.2.10 Reset control registers 218
2.2.10.1 Reset Control Register (SYSC_RSTCNTR) 218
2.2.10.2 User Reset Cause Register (SYSC_RSTCAUSEUR) 220
2.2.10.3 BootROM Reset Cause Register (SYSC_RSTCAUSEBT) 224
2.2.11 Slow RC Source Clock Timer registers 228
2.2.11.1 Slow RC SCT Trigger Register (SYSC_SRCSCTTRG) 228
2.2.11.2 Slow RC SCT Control Register (SYSC_SRCSCTCNTR) 230
2.2.11.3 Slow RC SCT Compare Prescaler Register (SYSC_SRCSCTCPR) 231
2.2.11.4 Slow RC SCT Status Register (SYSC_SRCSCTSTATR) 233
2.2.11.5 Slow RC SCT Interrupt Enable Register (SYSC_SRCSCTINTER) 235
2.2.11.6 Slow RC SCT Interrupt Clear Register (SYSC_SRCSCTICLR) 236
2.2.12 RC Source Clock Timer registers 237
2.2.12.1 RC SCT Trigger Register (SYSC_RCSCTTRG) 237
2.2.12.2 RC SCT Control Register (SYSC_RCSCTCNTR) 239
2.2.12.3 RC SCT Compare Prescaler Register (SYSC_RCSCTCPR) 240
2.2.12.4 RC SCT Status Register (SYSC_RCSCTSTATR) 242
2.2.12.5 RC SCT Interrupt Enable Register (SYSC_RCSCTINTER) 244
2.2.12.6 RC SCT Interrupt Clear Register (SYSC_RCSCTICLR) 245
2.2.13 Main Source Clock Timer Registers 246
2.2.13.1 Main SCT Trigger Register (SYSC_MAINSCTTRG) 246
2.2.13.2 Main SCT Control Register (SYSC_MAINSCTCNTR) 248
2.2.13.3 Main SCT Compare Prescaler Register (SYSC_MAINSCTCPR) 249
2.2.13.4 Main SCT Status Register (SYSC_MAINSCTSTATR) 251
2.2.13.5 Main SCT Interrupt Enable Register (SYSC_MAINSCTINTER) 253
2.2.13.6 Main SCT Interrupt Clear Register (SYSC_MAINSCTICLR) 254
2.2.14 Sub Source Clock Timer Registers 255
2.2.14.1 Sub SCT Trigger Register (SYSC_SUBSCTTRG) 255
2.2.14.2 Sub SCT Control Register (SYSC_SUBSCTCNTR) 257
2.2.14.3 Sub SCT Compare Prescaler Register (SYSC_SUBSCTCPR) 258

2.2.14.4 Sub SCT Status Register (SYSC_SUBSCTSTATR) 260
2.2.14.5 Sub SCT Interrupt Enable Register (SYSC_SUBSCTINTER) 262
2.2.14.6 Sub SCT Interrupt Clear Register (SYSC_SUBSCTICLR) 263
2.2.15 Clock output function configuration registers 264
2.2.15.1 Clock Output Function Configuration Register (SYSC_CKOTCFGR) 264
2.2.16 Special configuration registers 266
2.2.16.1 Special Configuration Register (SYSC_SPCCFGR) 266
2.2.16.2 RC Configuration Register (SYSC_RCCFGR) 270
2.2.16.3 Test 0 Register (SYSC_TESTR0) 271
2.2.16.4 Test 1 Register (SYSC_TESTR1) 272
2.2.16.5 Test 2 Register (SYSC_TESTR2) 273
2.2.17 Debugger registers 274
2.2.17.1 JTAG Detect Register (SYSC_JTAGDETECT) 274
2.2.17.2 JTAG Configuration Register (SYSC_JTAGCNFG) 275
2.2.17.3 JTAG Wakeup Register (SYSC_JTAGWAKEUP) 276
2.3 Operation of System Controller 277
2.3.1 Device mode 277
2.3.1.1 MODE selection 277
2.3.1.2 User mode 277
2.3.1.3 Board test mode 278
2.3.1.4 Boundary scan mode 278
2.3.2 Device state 278
2.3.2.1 RUN state 278
2.3.2.2 Power Saving State (PSS) 278
2.3.2.3 Device state switching 279
2.3.2.4 RUN to RUN state switching 279
2.3.2.5 RUN to PSS switching 281
2.3.2.6 PSS to RUN state switching 282
2.3.2.7 Device state profiles 284
2.3.2.8 RUN profile 285
2.3.2.9 PSS profile 286
2.3.2.10 Device state table 287
2.3.3 Reset and power up 290
2.3.3.1 Reset sources 290
2.3.3.2 Hard reset 290
2.3.3.3 Soft reset 291
2.3.3.4 Power domain reset 291
2.3.3.5 JTAG reset 291
2.3.3.6 Effect of reset sources 292
2.3.3.7 Reset, system clock and stabilization wait times 293
2.3.3.8 Stabilization wait time in case of a hard reset 293
2.3.3.9 Startup after power and external reset 293
2.3.3.10 Reset extension 294
2.3.3.11 Source clock stabilization 295
2.3.3.12 Start of program execution after hard reset 295
2.3.3.13 Power domain status 295
2.3.3.14 Operation of the low-voltage reset function 295
2.3.4 Power management 297
2.3.4.1 Overview 297
2.3.4.2 Power domain control 298
2.3.4.3 Power control mode 298
2.3.4.4 IO data retention at power OFF 300
2.3.4.5 Fake power down mode 301

2.3.5 Clock management 301
2.3.5.1 Clock overview 301
2.3.5.2 Clock generation block 303
2.3.5.3 Source clocks 303
2.3.5.4 Source clock usage 305
2.3.5.5 PLL interface 306
2.3.5.6 Oscillator connection 310
2.3.5.7 Clock selection and distribution block diagram 312
2.3.5.8 Clock selection 312
2.3.5.9 Clock distribution 313
2.3.6 Source Clock Timers 315
2.3.6.1 Overview of Source Clock Timer 315
2.3.6.2 Block diagram 316
2.3.7 Clock Supervisor 318
2.3.7.1 Block diagram 319
2.3.8 Clock output function 321
2.4 Notes on using System Controller 322
2.4.1 Register configuration 322
2.4.2 Device status handling 323
2.4.3 Error handling 323
2.4.4 Software based clock gearing 324
2.4.5 Notes on using Source Clock Timer 324
2.4.6 Notes on using Clock Supervisor 325

3. Real Time Clock 329

3.1 Outline of the Real Time Clock 329
3.2 Real Time Clock registers 333
3.2.1 Timer Control Register (RTC_WTCR) 335
3.2.2 Timer Status Register (RTC_WTSR) 339
3.2.3 Interrupt Status Register (RTC_WINS) 341
3.2.4 Interrupt Enable Register (RTC_WINE) 343
3.2.5 Interrupt Clear Register (RTC_WINC) 345
3.2.6 Sub-Second Register (RTC_WTBR) 347
3.2.7 Real Time Register (RTC_WRT) 349
3.2.8 Calibration Clock Counter Register (RTC_CNTCAL) 351
3.2.9 Calibration Clock Period Counter Register (RTC_CNTPCAL) 353
3.2.10 Calibration Duration Register (RTC_DURMW) 355
3.2.11 Calibration Trigger Register (RTC_CALTRG) 357
3.2.12 Debug Register (RTC_DEBUG) 358
3.3 Operation of the Real Time Clock 359
3.4 Cautions 364

4. Watchdog Timer 365

4.1 Outline of the Watchdog Timer 365
4.2 Watchdog Timer registers 367
4.2.1 Watchdog Protection Register (WDG_PROT) 370
4.2.2 Watchdog Counter Register (WDG_CNT) 371
4.2.3 Watchdog Reset Cause Register (WDG_RSTCAUSE) 372
4.2.4 Watchdog Trigger 0 Register (WDG_TRG0) 374
4.2.5 Watchdog Trigger 1 Register (WDG_TRG1) 375
4.2.6 Watchdog Interrupt Configuration Register (WDG_INT) 376
4.2.7 Watchdog Interrupt Clear Register (WDG_INTCLR) 378

4.2.8 Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG) 379
4.2.9 Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG) 380
4.2.10 Watchdog Lower Limit RUN Register (WDG_RUNLL) 381
4.2.11 Watchdog Upper Limit RUN Register (WDG_RUNUL) 382
4.2.12 Watchdog Lower Limit PSS Register (WDG_PSSLL) 383
4.2.13 Watchdog Upper Limit PSS Register (WDG_PSSUL) 384
4.2.14 Watchdog Reset Delay Counter Register (WDG_RSTDLY) 385
4.2.15 Watchdog Configuration Register (WDG_CFG) 386
4.3 Operation of the Watchdog Timer 389
4.3.1 Watchdog startup 389
4.3.2 Watchdog operation 391
4.4 Notes on using the Watchdog Timer 395

5. Security Checker 401

5.1 Outline of Security Checker 401
5.2 Security Checker registers 403
5.2.1 Flash Security registers 406
5.2.1.1 TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0~3) 406
5.2.1.2 EEFLASH Permission User Key Register (SCCFG_EEFPUSRKEY0~3) 407
5.2.2 System Interface registers 408
5.2.2.1 Security Control Register (SCCFG_CTRL) 408
5.2.2.2 Security Status Register 0 (SCCFG_STAT0) 409
5.2.2.3 Security Status Register 1 (SCCFG_STAT1) 412
5.2.2.4 Security Status Register 2 (SCCFG_STAT2) 414
5.2.3 Device Security registers 416
5.2.3.1 Security Key Register (SCCFG_SECKEY0~3) 416
5.2.4 General registers 417
5.2.4.1 Module ID Register (SCCFG_MODID) 417
5.2.4.2 Security Checker Unlock Register (SCCFG_UNLCK) 418
5.2.4.3 General Purpose Register (SCCFG_GPREG0~1) 419
5.3 Operation of Security Checker 420
5.4 Notes on using Security Checker 423

6. Memory Protection Unit for AXI 427

6.1 Outline of MPU AXI 427
6.2 MPU AXI registers 429
6.2.1 MPU AXI Control Register (MPUXn_CTRL0) 432
6.2.2 MPU AXI NMI Enable Register (MPUXn_NMIEN) 435
6.2.3 MPU AXI Write Error Control Register (MPUXn_WERRC) 436
6.2.4 MPU AXI Write Error Address Register (MPUXn_WERRA) 437
6.2.5 MPU AXI Read Error Control Register (MPUXn_RERRC) 438
6.2.6 MPU AXI Read Error Address Register (MPUXn_RERRA) 439
6.2.7 MPU AXI Region Control Registers (MPUXn_CTRL1~8) 440
6.2.8 MPU AXI Start Address Registers (MPUXn_SADDR1~8) 442
6.2.9 MPU AXI End Address Registers (MPUXn_EADDR1~8) 443
6.2.10 MPU AXI Unlock Register (MPUXn_UNLOCK) 444
6.2.11 MPU AXI Module ID Register (MPUXn_MID) 445
6.3 Operation of MPU AXI 446
6.4 Notes on using MPU AXI 451

7. Memory Protection Unit for AHB 453

7.1 Outline of the MPU AHB 453
7.2 MPU AHB registers....455
7.2.1 MPU AHB Control Register (MPUHn_CTRL0)....458
7.2.2 MPU AHB NMI Enable Register (MPUHn_NMIEN) 460
7.2.3 MPU AHB Memory Error Control Register (MPUHn_MERRC)....461
7.2.4 MPU AHB Memory Error Address Register (MPUHn_MERRA)....462
7.2.5 MPU AHB Region Control Registers (MPUHn_CTRL1~8) 463
7.2.6 MPU AHB Start Address Registers (MPUHn_SADDR1~8) 465
7.2.7 MPU AHB End Address Registers (MPUHn_EADDR1~8)....466
7.2.8 MPU AHB Unlock Register (MPUHn_UNLOCK)....467
7.2.9 MPU AHB Module ID Register (MPUHn_MID)....468
7.3 Operation of the MPU AHB....469
7.4 Notes on using the MPU AHB 472

8. Programmable CRC 475

8.1 Outline of the Programmable CRC 475
8.2 Programmable CRC registers....477
8.2.1 CRC Polynomial Register (CRCn_POLY)....478
8.2.2 CRC Seed Register (CRCn_SEED)....479
8.2.3 CRC Final XOR Register (CRCn_FXOR) 480
8.2.4 CRC Configuration Register (CRCn_CFG)....481
8.2.5 CRC Write Register (CRCn_WR) 484
8.2.6 CRC Read Register (CRCn_RD)....485
8.3 Operation of the Programmable CRC....486
8.3.1 CRC operation flowcharts 487
8.3.2 CRC input data and checksum calculation flow 490
8.3.3 CRC calculation example....494

9. Tightly Coupled Flash 495

9.1 Outline of the TCFLASH 495
9.2 TCFLASH registers....497
9.2.1 Flash Configuration Protection Key Register (TCFCFG _FCPROTKEY) 499
9.2.2 Flash Configuration Register (TCFCFG_FCFGR) 500
9.2.3 Flash ECC Control Register (TCFCFG_FECCCTRL)....502
9.2.4 Flash Data Bit Error Injection Register (TCFCFG_FDATEIR)....503
9.2.5 Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR) 504
9.2.6 Flash Interrupt Control Register (TCFCFG_FICTRLn) 505
9.2.7 Flash Status Register (TCFCFG_FSTATn)....507
9.2.8 Flash SEC Interrupt Register (TCFCFG_FSECIR)....509
9.2.9 Flash ECC Error Address Register (TCFCFG_FECCEAR) 510
9.2.10 Flash Interface Module Identification Register (TCFCFG_FMIDR)....511
9.2.11 Flash CAM Output Lower Register (TCFCFG_FCAMLRn)....512
9.2.12 Flash CAM Output Upper Register (TCFCFG_FCAMHRn) 513
9.3 Operation of the TCFLASH....514
9.4 Starting the Flash memory automatic algorithm 523
9.5 Confirming the automatic algorithm execution state....526
9.5.1 Data polling flags (DQ[7,15])....528
9.5.2 Toggle bit flags (DQ[6,14]) 529
9.5.3 Timing-limit-exceeded flags (DQ[5,13])....530
9.5.4 Sector erase timer flags (DQ[3,11])....531
9.5.5 Toggle bit 2 flags (DQ[2,10]) 532

9.6 Writing to and erasing Flash memory 533
9.6.1 Setting read/reset state 534
9.6.2 Writing data by submitting the write command sequence 535
9.6.3 Erasing all data (macro erase) 538
9.6.4 Erasing optional data (sector erase) 539
9.6.5 Suspending sector erase 542
9.6.6 Sector erase resume 544
9.7 Flash security 545
9.8 Notes on using Flash memory 546

10. EEPROM Emulation Flash 549

10.1 Outline of the EEPROM Emulation Flash 549
10.2 EEPROM Emulation Flash Memory Interface registers 551
10.2.1 Configuration Protection Key Register (EEFCFG_CPR) 553
10.2.2 Configuration Register (EEFCFG_CR) 554
10.2.3 ECC Control Register (EEFCFG_ECR) 556
10.2.4 Write Command Sequencer Configuration Register (EEFCFG_WCR) 557
10.2.5 Write Command Sequencer Status Register (EEFCFG_WSR) 558
10.2.6 Data Bit Error Injection Register (EEFCFG_DBEIR) 559
10.2.7 ECC Bit Error Injection Register (EEFCFG_EEIR) 560
10.2.8 Write Mode Enable Register (EEFCFG_WMER) 561
10.2.9 Interrupt Control Register (EEFCFG_ICR) 562
10.2.10 Status Register (EEFCFG_SR) 564
10.2.11 SEC Interrupt Register (EEFCFG_SECIR) 566
10.2.12 ECC Error Address Register (EEFCFG_EEAR) 568
10.2.13 Module Identification Register (EEFCFG_MIR) 569
10.2.14 Extra Mode Enable Register (EEFCFG_EMENR) 570
10.2.15 Flash CAM Output Lower Register (EEFCFG_FCAMLR) 571
10.2.16 Flash CAM Output Higher Register (EEFCFG_FCAMHR) 572
10.3 Operation of the EEPROM Emulation Flash Interface 573
10.4 Starting the EEPROM Emulation Flash memory automatic algorithm 580
10.5 Notes on using Flash memory 582
10.6 Notes on enabling ECC 583

11. EEPROM Emulation Flash and SHE 587

11.1 Outline of the EEPROM Emulation Flash 587
11.2 EEPROM Emulation Flash Memory Interface registers 589
11.2.1 Configuration Protection Key Register (EEFCFG_CPR) 592
11.2.2 Configuration Register (EEFCFG_CR) 593
11.2.3 ECC Control Register (EEFCFG_ECR) 595
11.2.4 Write Command Sequencer Configuration Register (EEFCFG_WCR) 596
11.2.5 Write Command Sequencer Status Register (EEFCFG_WSR) 597
11.2.6 Data Bit Error Injection Register (EEFCFG_DBEIR) 598
11.2.7 ECC Bit Error Injection Register (EEFCFG_EEIR) 599
11.2.8 Interrupt Control Register (EEFCFG_ICR) 600
11.2.9 Status Register (EEFCFG_SR) 601
11.2.10 SEC Interrupt Register (EEFCFG_SECIR) 603
11.2.11 ECC Error Address Register (EEFCFG_EEAR) 605
11.2.12 Module Identification Register (EEFCFG_MIR) 606
11.2.13 Extra Mode Enable Register (EEFCFG_EMENR) 607
11.2.14 Flash Register (EEFCFG_FCAMLR) 608
11.2.15 Flash Register (EEFCFG_FCAMHR) 609

11.2.16Sequencer Write Mode Register (EEFCFG_SEQWM) 610
11.2.17Sequencer Command Mode Register (EEFCFG_SEQCM) 611
11.2.18Arbitration Error Register (EEFCFG_ARBERR) 613
11.2.19Read Arbitration Error Register (EEFCFG_ARBCLR) 614
11.2.20Bus Error Response Register (EEFCFG_BERR) 615
11.2.21Bus Error Response Clear Register (EEFCFG_BERRCLR) 616
11.3 Operation of the EEPROM Emulation Flash Interface 618
11.4 Notes on using Flash memory 625
11.5 Support for the Secure Hardware Extension (SHE) 626

12. TCM RAM Interface 629

12.1 Outline of the TCMRAM_IF 629
12.2 TCMRAM_IF registers 631
12.2.1 TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0) 632
12.2.2 TCMRAM_IF Configuration Register 1 (TRCFG_TCMCFG1) 634
12.2.3 TCMRAM_IF UNLOCK Register (TRCFG_TCMUNLOCK) 635
12.3 Operation of the TCMRAM_IF 636
12.4 TCMRAM_IF flowchart 637

13. SRAM Interface 639

13.1 Outline of the SRAM_IF 639
13.2 SRAM_IF registers 641
13.2.1 SRAM_IF Configuration Register 0 (SRCFG_CFG0) 642
13.2.2 SRAM_IF Configuration Register 1 (SRCFG_CFG1) 644
13.2.3 SRAM_IF Configuration Register 2 (SRCFG_CFG2) 645
13.2.4 SRAM_IF Unlock/Lock Key Register (SRCFG_KEY) 646
13.2.5 SRAM_IF Error Flag Register (SRCFG_ERRFLG) 647
13.2.6 SRAM_IF Interrupt Enable Register (SRCFG_INTE) 648
13.2.7 SRAM_IF ECC Enable Register (SRCFG_ECCE) 649
13.2.8 SRAM_IF Error Address Register (SRCFG_ERRADR) 650
13.2.9 SRAM_IF Module Identification Register (SRCFG_MID) 651
13.3 Operation of the SRAM_IF 652
13.4 How to use the SRAM_IF 654

14. Resource Input Configuration 655

14.1 Outline of the Resource Input Configuration 655
14.2 Resource Input Configuration registers 657
14.2.1 Resource Input Configuration Registers (RICFG_RESnIN) 658
14.3 Notes on using the Resource Input Configuration 661
14.4 Special connections 662

15. Boot ROM Hardware Interface 667

15.1 Outline of Boot ROM Hardware Interface 667
15.2 Boot ROM Hardware Interface registers 668
15.2.1 Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK) 670
15.2.2 Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG) 671
15.2.3 Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT) 673
15.2.4 Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT) 674
15.2.5 Prefetch Abort Vector Register - Inactive Set (EXCFG_PABORTINACT) 675
15.2.6 Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT) 676
15.2.7 Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT) 677
15.2.8 Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT) 678

15.2.9 Supervisor Call Vector Register - Active Set (EXCFG_SVCACT) 679
15.2.10 Prefetch Abort Vector Register - Active Set (EXCFG_PABORTACT) 680
15.2.11 Data Abort Vector Register - Active Set (EXCFG_DABORTACT) 681
15.2.12 Interrupt Vector Register - Active Set (EXCFG_IRQACT) 682
15.3 Operation of Boot ROM Hardware Interface 683
15.4 Boot ROM Hardware Interface flowchart 685

16. Boot ROM Software Interface 687

16.1 Outline of the Boot ROM Software Interface 687
16.2 Boot ROM markers in Flash memory 689
16.2.1 SHE Secure Boot Mode Marker (BDR_SBMM) 698
16.2.2 SHE Secure Boot Size Marker (BDR_SBSM) 699
16.2.3 Debugger Waiting Enable Marker (BDR_DWEM) 700
16.2.4 Alternative Boot Vector Marker (BDR_ABVM) 701
16.2.5 Alternative Boot Vector Enable Marker (BDR_ABVEM) 702
16.2.6 Primary Device Security Enable Marker (MSDR_PDSEM)/ Secondary Device Security Enable Marker (MSDR_SDSEM) 703
16.2.7 Primary Device Security Key Marker (MSDR_PDSKM0~3)/ Secondary Device Security Key Marker (MSDR_SDSKM0~3) 705
16.2.8 Primary FFA Key Marker (MSDR_PFFAKM0~3)/Secondary FFA Key Marker (MSDR_SFFAKM0~3) 708
16.2.9 Primary Device Security Control Marker (MSDR_PCTRLM)/Secondary Device Security Control Marker (MSDR_SCTRLM) 710
16.2.10 Primary Configuration Enable Marker (MSDR_PCEM)/ Secondary Configuration Enable Marker (MSDR_SCEM) 715
16.2.11 TCFLASH Primary Link Key Marker (TCSDRn_PLKM0~3)/ TCFLASH Secondary Link Key Marker (TCSDRn_SLKM0~3) 717
16.2.12 TCFLASH Primary Permission Key Marker (TCSDRn_PFPKM0~3)/TCFLASH Secondary Permission Key Marker (TCSDRn_SFPKM0~3) 719
16.2.13 TCFLASH Permission Set Marker (TCSDRn_PFPS0M0~15/TCSDRn_SFPS0M0~15 and TCSDRn_PFPS1M0~15/TCSDRn_SFPS1M0~15) 721
16.2.14 EEFLASH Link Key Marker (EESDR0_LKM0~3) 735
16.2.15 EEFLASH Permission Key Marker (EESDR_FPKM0~3) 736
16.2.16 EEFLASH Permission Set Marker (EESDR_FPS0M0~3 and EESDR_FPS1M0~3) 737
16.3 SDR Validation Status 744
16.3.1 SDR Validation Status in GPREG1 (SCCFG_GPREG1) 745
16.4 Operation of the Boot ROM 748

17. Interrupt Controller 753

17.1 Outline of Interrupt Controller 753
17.2 Interrupt Controller registers 755
17.2.1 IUNIT NMI Vector Address Status Register (IRQn_NMIVAS) 779
17.2.2 IUNIT NMI Status Register (IRQn_NMIST) 780
17.2.3 IUNIT IRQ Vector Address Status Register (IRQn_IRQVAS) 781
17.2.4 IUNIT IRQ Status Register (IRQn_IRQST) 782
17.2.5 IUNIT NMI Vector Address Register (IRQn_NMIVA0~31) 783
17.2.6 IUNIT IRQ Vector Address Register (IRQn_IRQVA0~511) 784
17.2.7 IUNIT NMI Priority Level Register (IRQn_NMIPL0) 785
17.2.8 IUNIT NMI Priority Level Register (IRQn_NMIPL1~7) 786
17.2.9 IUNIT IRQ Priority Level Register (IRQn_IRQPL0~127) 787
17.2.10 IUNIT NMI Set Register (IRQn_NMIS) 788

17.2.11IUNIT NMI Reset Register (IRQn_NMIR) 789
17.2.12IUNIT NMI Software Interrupt Status Register (IRQn_NMISIS)....790
17.2.13IUNIT IRQ Set Register (IRQn_IRQS0~15)....791
17.2.14IUNIT IRQ Reset Register (IRQn_IRQR0~15) 792
17.2.15IUNIT IRQ Software Interrupt Status Register (IRQn_IRQSIS0~15)....793
17.2.16IUNIT IRQ Channel Enable Set Register (IRQn_IRQCES0~15) 794
17.2.17IUNIT IRQ Channel Enable Clear Register (IRQn_IRQCEC0~15)....795
17.2.18IUNIT IRQ Channel Enable Register (IRQn_IRQCE0~15)....796
17.2.19IUNIT NMI Hold Clear Register (IRQn_NMIHC)....797
17.2.20IUNIT NMI Hold Status Register (IRQn_NMIHS) 798
17.2.21IUNIT IRQ Hold Clear Register (IRQn_IRQHC) 799
17.2.22IUNIT IRQ Hold Status Register (IRQn_IRQHS0~15)....800
17.2.23IUNIT IRQ Priority Level Mask Register (IRQn_IRQPLM)....801
17.2.24IUNIT Control and Status Register (IRQn_CSR)....802
17.2.25IUNIT Nesting Level Status Register (IRQn_NESTL)....803
17.2.26IUNIT NMI Raw Status Register (IRQn_NMIRS)....804
17.2.27IUNIT NMI Preprocessed Status Register (IRQn_NMIPS)....805
17.2.28IUNIT IRQ Raw Status Register (IRQn_IRQRS0~15)....806
17.2.29IUNIT IRQ Preprocessed Status Register (IRQn_IRQPS0~15) 807
17.2.30IUNIT Unlock Register (IRQn_UNLOCK) 808
17.2.31IUNIT Module Identification Register (IRQn_MID)....809
17.2.32IUNIT ECC Error Interrupt Register (IRQn_EEI) 810
17.2.33IUNIT ECC Address Number Register (IRQn_EAN) 812
17.2.34IUNIT ECC Test Register (IRQn_ET)....813
17.2.35IUNIT ECC Error Bits Register (IRQn_EEB0 ~ 1) 814
17.2.36IUNIT ECC Error Bits Register (IRQn_EEB2) 815
17.3 Interrupt Controller stages 816
17.4 Notes for software....818
17.5 Operations flowchart....819

18. Port Pin Configuration 829

18.1 Outline of Port Pin Configuration 829
18.2 Port Pin Configuration registers....831
18.2.1 Pin Configuration Register (PPC_PCFGRijj) 832
18.3 Notes on using Port Pin Configuration....834

19. External Interrupt Capture Unit 837

19.1 Outline of the External Interrupt Capture Unit (EICU)....837
19.2 EICU registers 839
19.2.1 Configuration Register (EICUn_CNFGR)....840
19.2.2 External Interrupt Pin Enable Register (EICUn_IRENR)....843
19.2.3 Sample Registers 0~7 (EICUn_SPLR0~7) 844
19.3 Operation of the EICU 845
19.4 Notes on using the EICU 847

20. Timing Protection Unit 849

20.1 Outline of TPU 849
20.2 TPU registers....851
20.2.1 TPU Unlock Register (TPUn_UNLOCK)....853
20.2.2 TPU Lock Status Register (TPUn_LST)....854
20.2.3 TPU Configuration Register (TPUn_CFG)....855
20.2.4 TPU Timer Interrupt Request Register (TPUn_TIR) 857

20.2.5 TPU Timer Status Register (TPUn_TST) 858
20.2.6 TPU Timer Interrupt Enable Register (TPUn_TIE) 859
20.2.7 TPU Module ID Register (TPUn_MID) 860
20.2.8 TPU Timer Control Register 0 (TPUn_TCN00~07) 861
20.2.9 TPU Timer Control Register 1 (TPUn_TCN10~17) 863
20.2.10 TPU Timer Current Count Register (TPUn_TCC0) 865
20.3 Operation of TPU 866
20.4 TPU flowchart 868

21. Peripheral Protection Unit 871

21.1 Outline of Peripheral Protection Unit 871
21.2 PPU registers 873
21.2.1 PPU Peripheral Read Attribute Register (PPUn_PR0~15) 878
21.2.2 PPU Peripheral Read Attribute Set Register (PPUn_PRS0~15) 879
21.2.3 PPU Peripheral Read Attribute Clear Register (PPUn_PRC0~15) 880
21.2.4 PPU Peripheral Access Attribute Register (PPUn_PA0~15) 881
21.2.5 PPU Peripheral Access Attribute Set Register (PPUn_PAS0~15) 882
21.2.6 PPU Peripheral Access Attribute Clear Register (PPUn_PAC0~15) 883
21.2.7 PPU GPIO Access Attribute Register (PPUn_GA0~15) 884
21.2.8 PPU GPIO Access Attribute Set Register (PPUn_GAS0~15) 885
21.2.9 PPU GPIO Access Attribute Clear Register (PPUn_GAC0~15) 886
21.2.10 PPU Status Register (PPUn_ST) 887
21.2.11 PPU Control Register (PPUn_CTR) 888
21.2.12 PPU Unlock Register (PPUn_UNLOCK) 890
21.3 Operation of PPU 891
21.4 Notes on using PPU 893

22. Bus Error Collection Unit 895

22.1 Outline of Bus Error Collection Unit (BECU) 895
22.2 BECU registers 896
22.2.1 BECU Control Register - L (BECUn_CTRL) 898
22.2.2 BECU Control Register - H (BECUn_CTRH) 899
22.2.3 BECU Address Register - L (BECUn_ADDRL) 900
22.2.4 BECU Address Register - H (BECUn_ADDRH) 901
22.2.5 BECU Data Register - LL (BECUn_DATALL) 902
22.2.6 BECU Data Register - LH (BECUn_DATALH) 903
22.2.7 BECU Data Register - HL (BECUn_DATAHL) 904
22.2.8 BECU Data Register - HH (BECUn_DATAHH) 905
22.2.9 BECU Master ID Register (BECUn_MASTERID) 906
22.2.10 BECU Module ID Register - L (BECUn_MIDL) 907
22.2.11 BECU Module ID Register - H (BECUn_MIDH) 908
22.2.12 BECU NMI Enable Register - (BECUn_NMIEN) 909
22.3 Operation of BECU 910
22.4 Notes on using the BECU 911

23. RETENTIONRAM 913

23.1 Outline of the RETENTIONRAM 913
23.2 RETENTIONRAM registers 915
23.2.1 RETENTIONRAM Unlock Register (RRCFG_UNLOCKR) 916
23.2.2 RETENTIONRAM Configuration and Status Register (RRCFG_CSR) 918
23.2.3 RETENTIONRAM ECC Error Address Number Register (RRCFG_EAN) 921
23.2.4 RETENTIONRAM Error Mask Register 0 (RRCFG_ERRMSKR0) 922

23.2.5 RETENTIONRAM Error Mask Register 1 (RRCFG_ERRMSKR1) 923
23.2.6 RETENTIONRAM ECC Enable Register (RRCFG_ECCEN) 924
23.3 Operation of the RETENTIONRAM 925
23.4 Notes on using the RETENTIONRAM 927

24. External Interrupts 931

24.1 Outline of the External Interrupts module 931
24.2 External Interrupts module registers 933
24.2.1 External Interrupt Enable Register (EICn_ENIR) 935
24.2.2 External Interrupt Enable Set Register (EICn_ENISR) 936
24.2.3 External Interrupt Enable Clear Register (EICn_ENICR) 937
24.2.4 External Interrupt Request Register (EICn_EIRR) 938
24.2.5 External Interrupt Request Clear Register (EICn_EIRCR) 939
24.2.6 Noise Filter Enable Register (EICn_NFER) 940
24.2.7 Noise Filter Enable Set Register (EICn_NFESR) 941
24.2.8 Noise Filter Enable Clear Register (EICn_NFECR) 942
24.2.9 External Interrupt Level Register (EICn_ELVR0~3) 943
24.2.10 Non-Maskable Interrupt Register (EICn_NMIR) 946
24.2.11 DMA Request Enable Register (EICn_DRER) 947
24.2.12 DMA Request Enable Set Register (EICn_DRESR) 948
24.2.13 DMA Request Enable Clear Register (EICn_DRECR) 949
24.2.14 DMA Request Flag Register (EICn_DRFR) 950
24.3 Operation of the External Interrupts module 951
24.4 Notes on using the External Interrupts module 953

25. Sound Generator 955

25.1 Outline of the Sound Generator 955
25.2 Sound Generator registers 957
25.2.1 Sound Generator Control Register 0 (SGn_CR0) 959
25.2.2 Sound Generator Control Register 1 (SGn_CR1) 962
25.2.3 Sound Generator Extended Control Reload Register (SGn_ECRL) 964
25.2.4 Sound Generator Frequency Data Reload Register (SGn_FRL) 967
25.2.5 Sound Generator Amplitude Status Register (SGn_AR) 968
25.2.6 Sound Generator Amplitude Data Reload Register (SGn_ARL) 969
25.2.7 Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL) 970
25.2.8 Sound Generator Target Amplitude Data Reload Register (SGn_TARL) 972
25.2.9 Sound Generator Tone Output Number Reload Register (SGn_NRL) 973
25.2.10 Sound Generator DMA Transfer Update Enable Register (SGn_DER) 974
25.2.11 Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR) 977
25.3 Operation of the Sound Generator 979

26. 16-Bit I/O Timer 991

26.1 Outline of the 16-bit I/O Timer 991
26.2 Free Running Timer (FRT) 993
26.2.1 16-bit FRT registers 995
26.2.1.1 Data Register (FRTn_TCDT) 996
26.2.1.2 Compare Clear Buffer Register (FRTn_CPCLRB) 997
26.2.1.3 Compare Clear Register (FRTn_CPCLR) 998
26.2.1.4 Control Status Register (FRTn_TCCS) 999
26.2.1.5 Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK) 1001
26.2.1.6 Extended Control Status Register (FRTn_ETCCS) 1003

26.2.1.7 Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS) 1005
26.2.1.8 DMA Configuration Register (FRTn_DMACFG) 1007
26.2.2 Operation of 16-bit FRT 1008
26.3 Output Compare Unit (OCU) 1012
26.3.1 16-bit OCU registers 1014
26.3.1.1 Output Compare Register (OCUn_OCCP0/OCUn_OCCP1) 1016
26.3.1.2 Output Compare Buffer Registers (OCUn_OCCPB0/ OCUn_OCCPB1) 1018
26.3.1.3 Output Compare Down Buffer Registers (OCUn_OCCPBD0/OCUn_OCCPBD1) 1020
26.3.1.4 Control Register (OCUn_OCS01) 1022
26.3.1.5 Control Clear Register (OCUn_OCSC01) 1025
26.3.1.6 Control Set Register (OCUn_OCSS01) 1027
26.3.1.7 Status Register (OCUn_OSR01) 1029
26.3.1.8 Status Clear Register (OCUn_OSCR01) 1030
26.3.1.9 Extended Output Compare Control Status Register(OCUn_EOCS01) 1031
26.3.1.10 Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01) 1035
26.3.1.11 Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01) 1036
26.3.1.12 Debug Configuration Register (OCUn_DEBUG01) 1037
26.3.1.13 DMA Configuration Register (OCUn_DMACFG01) 1038
26.3.1.14 Compare Mode Control Register (OCUn_OCMCR01) 1039
26.3.2 Operation of 16-bit Output Compare Unit 1042
26.4 Input Capture Unit (ICU) 1052
26.4.1 Input Capture Unit registers 1053
26.4.1.1 Input Capture Data Register 0 (ICUn_IPC0) 1054
26.4.1.2 Input Capture Data Register 1 (ICUn_IPC1) 1055
26.4.1.3 Input Capture Clear Register (ICUn_ICC01) 1056
26.4.1.4 Input Capture Edge Register and Control Status Register (ICUn_ICEICS01) 1058
26.4.1.5 DMA Configuration Register (ICUn_DMACFG01) 1061
26.4.1.6 Debug Register (ICUn_DEBUG01) 1062
26.4.2 Operation of the 16-bit input capture 1063
26.4.3 Application of 16-bit I/O Timer 1065

27. Programmable Pulse Generator 1067

27.1 Outline of the Programmable Pulse Generator 1067
27.2 Programmable Pulse Generator registers 1071
27.2.1 PPG Control Status Register (PPGn_PCN) 1074
27.2.2 Interrupt Flag Clear Register (PPGn_IRQCLR) 1076
27.2.3 Software Trigger Activation Register (PPGn_SWTRIG) 1077
27.2.4 Output Enable Register (PPGn_OE) 1078
27.2.5 Timer Enable Operation Register (PPGn_CNTEN) 1079
27.2.6 Output Mask and Polarity Selection Register (PPGn_OPTMSK) 1080
27.2.7 Ramp Configuration Register (PPGn_RMPCFG) 1081
27.2.8 Start Delay Mode Register (PPGn_STRD) 1083
27.2.9 PPG Trigger Clear Flag Register (PPGn_TRIGCLR) 1084
27.2.10 Extended PPG Control Status Register 1 (PPGn_EPCN1) 1085
27.2.11 Extended PPG Control Status Register 2 (PPGn_EPCN2) 1087
27.2.12 General Control Register 1 (PPGn_GCN1) 1090
27.2.13 General Control Register 3 (PPGn_GCN3) 1092

27.2.14 General Control Register 4 (PPGn_GCN4) 1093
27.2.15 General Control Register 5 (PPGn_GCN5) 1095
27.2.16 PPG Cycle Setting Register (PPGn_PCSR) 1096
27.2.17 PPG Duty Setting Register (PPGn_PDUT) 1097
27.2.18 PPG Timer Register (PPGn_PTMR) 1099
27.2.19 PPG Start Delay Register (PPGn_PSDR) 1100
27.2.20 PPG Timing Point Capture Register (PPGn_PTPC) 1101
27.2.21 PPG End Duty Register (PPGn_PEDR) 1102
27.2.22 PPG DMA Configuration Register (PPGn_DMACFG) 1104
27.2.23 PPG Debug Enable Register (PPGn_DEBUG) 1105
27.2.24 Group Control Register (PPGGRPp_GCTRL) 1106
27.2.25 PPG Global Control Register (PPGGLCg_GCNR) 1107
27.3 Operation of the Programmable Pulse Generator 1108
27.4 Cautions 1115

28. 32-Bit Reload Timer 1117

28.1 Outline of the 32-bit Reload Timer 1117
28.2 32-bit Reload Timer registers 1119
28.2.1 DMA Configuration Register (RLTn_DMACFG) 1120
28.2.2 Timer Control Status Register (RLTn_TMCSR) 1121
28.2.3 32-bit Reload Register (RLTn_TMRLR) 1126
28.2.4 32-bit Timer Register (RLTn_TMR) 1127
28.3 Internal clock and external event counter operation of the 32-bit Reload Timer 1128
28.4 Underflow operation of the 32-bit Reload Timer 1130
28.5 Output functions of the the 32-bit Reload Timer 1131
28.6 Counter operation state 1132
28.7 DMA operation 1133

29. Stepper Motor Controller 1135

29.1 Outline of the Stepper Motor Controller 1135
29.2 Stepper Motor Controller registers 1136
29.2.1 PWM Control Register (SMCn_PWC) 1138
29.2.2 PWM Control Set Register (SMCn_PWCS) 1140
29.2.3 PWM Control Clear Register (SMCn_PWCC) 1141
29.2.4 PWM Compare Registers (SMCn_PWC1, SMCn_PWC2) 1142
29.2.5 PWM Selection Register (SMCn_PWS) 1144
29.2.6 PWM Selection Set Register (SMCn_PWSS) 1147
29.2.7 SMC Trigger Delay Register (SMCn_PTRGDL) 1148
29.2.8 SMC Debug Register (SMCn_DEBUG) 1149
29.2.9 SMC Trigger Selection Register (SMCTGg_PTRGS) 1150
29.2.10 SMC Trigger Register (SMCTGg_PTRG) 1153
29.3 Operation of the Stepper Motor Controller 1155
29.3.1 Operation of the PWM Pulse Generator 1156
29.3.2 Operation of SMC Trigger Generator 1159
29.4 Notes on using the Stepper Motor Controller 1161

30. CAN Controller 1163

30.1 Overview of CAN 1163
30.2 CAN registers 1164
30.2.1 CAN Device Related Register 1169
30.2.1.1 CAN Output Enable Register (CANn_COER) 1169
30.2.2 CAN Protocol Related Registers 1170

30.2.2.1 CAN Control Register (CANn_CTRLR) 1170
30.2.2.2 Status Register (CANn_STATR) 1172
30.2.2.3 Error Counter (CANn_ERRCNT) 1174
30.2.2.4 Bit Timing Register (CANn_BTR) 1175
30.2.2.5 Test Register (CANn_TESTR) 1176
30.2.2.6 BRP Extension Register (CANn_BRPER) 1177
30.2.3 Message Interface Register Sets 1178
30.2.3.1 IFx Command Request Registers (CANn_IFxCREQ) 1179
30.2.3.2 IFx Command Mask Register (CANn_IFxCMSK) 1181
30.2.3.3 IFx Mask Registers (CANn_IFxMSK1, CANn_IFxMSK2) 1184
30.2.3.4 IFx Arbitration Registers (CANn_IF1ARB1, CANn_IF2ARB2) 1186
30.2.3.5 IFx Message Control Register (CANn_IF1MCTR) 1188
30.2.3.6 IFx Data A and Data B Registers (IFxDTAn, IFxDTBn) 1190
30.2.4 Message Handler Registers 1192
30.2.4.1 Interrupt Register (INTRn) 1192
30.2.4.2 Transmission Request Registers (TREQR1n, TREQR2n) 1194
30.2.4.3 New Data Registers (NEWDT1n, NEWDT2n) 1197
30.2.4.4 Interrupt Pending Registers (INTPND1n, INTPND2n) 1200
30.2.4.5 Message Valid Registers (MSGVAL1n, MSGVAL2n) 1203
30.2.5 Debug Register (DEBUGn) 1206
30.3 Message Object in the Message Memory 1207
30.4 Functional description 1210
30.5 CAN application 1214

31. LIN-USART 1223

31.1 Overview of the LIN-USART 1223
31.1.1 Block diagram of LIN-USART 1226
31.2 LIN-USART registers 1228
31.2.1 Serial Mode Register (USARTn_SMR) 1231
31.2.2 Serial Control Register (USARTn_SCR) 1233
31.2.3 Serial Mode Set Register (USARTn_SMSR) 1235
31.2.4 Serial Control Set Register (USARTn_SCSR) 1236
31.2.5 Serial Control Clear Register (USARTn_SCCR) 1237
31.2.6 Transmission Data Register (USARTn_TDR) 1238
31.2.7 Serial Status Register (USARTn_SSR) 1240
31.2.8 Reception Data Register (USARTn_RDR) 1245
31.2.9 Serial Status Set Register (USARTn_SSSR) 1247
31.2.10 Serial Status Clear Register (USARTn_SSCR) 1248
31.2.11 Extended Communication Control Register (USARTn_ECCR) 1249
31.2.12 Extended Status/Control Register (USARTn_ESCR) 1251
31.2.13 Extended Communication Control Set Register (USARTn_ECCSR) 1254
31.2.14 Extended Status/Control Set Register (USARTn_ESCSR) 1255
31.2.15 Extended Communication Control Clear Register (USARTn_ECCCR) 1256
31.2.16 Extended Status/Control Clear Register (USARTn_ESCCR) 1257
31.2.17 Extended Serial Interrupt Register (USARTn_ESIR) 1259
31.2.18 Extended Interrupt Enable Register (USARTn_EIER) 1262
31.2.19 Extended Serial Interrupt Set Register (USARTn_ESISR) 1264
31.2.20 Extended Interrupt Enable Set Register (USARTn_EIESR) 1265
31.2.21 Extended Serial Interrupt Clear Register (USARTn_ESICR) 1266
31.2.22 Extended Interrupt Enable Clear Register (USARTn_EIECR) 1268
31.2.23 Extended Feature Enable Register - L (USARTn_EFERL) 1269
31.2.24 Extended Feature Enable Register - H (USARTn_EFERH) 1271
31.2.25 Reception FIFO Control Register (USARTn_RFCR) 1273

31.2.26 Transmission FIFO Control Register (USARTn_TFCR) 1275
31.2.27 Reception FIFO Control Set Register (USARTn_RFCSR) 1277
31.2.28 Transmission FIFO Control Set Register (USARTn_TFCSR) 1278
31.2.29 Reception FIFO Control Clear Register (USARTn_RFCCR) 1279
31.2.30 Transmission FIFO Control Clear Register (USARTn_TFCCR) 1280
31.2.31 Reception FIFO Status Register (USARTn_RFSR) 1281
31.2.32 Transmission FIFO Status Register (USARTn_TFSR) 1282
31.2.33 Extended Status Register (USARTn_ESR) 1283
31.2.34 Extended Status Clear Register (USARTn_ESCLR) 1285
31.2.35 Baud Rate Generation Reload Register (USARTn_BGRLL) 1286
31.2.36 Baud Rate Generation Reload Register(USARTn_BGRLM) 1287
31.2.37 Baud Rate Generation Reload Register (USARTn_BGRLH) 1288
31.2.38 Baud Rate Generation Register (USARTn_BGRL) 1289
31.2.39 Baud Rate Generation Register (USARTn_BGRM) 1290
31.2.40 Baud Rate Generation Register (USARTn_BGRH) 1291
31.2.41 Serial Transmit DMA Configuration Register (USARTn_STXDR) 1292
31.2.42 Serial Receive DMA Configuration Register (USARTn_SRXDR) 1293
31.2.43 Serial Transmit DMA Configuration Set Register (USARTn_STXDSR) 1294
31.2.44 Serial Receive DMA Configuration Set Register (USARTn_SRXDSR) 1295
31.2.45 Serial Transmit DMA Configuration Clear Register (USARTn_STXDCR) 1296
31.2.46 Serial Receive DMA Configuration Clear Register (USARTn_SRXDCR) 1297
31.2.47 Debug Register (USARTn_DEBUG) 1298
31.2.48 Unused Registers 1299
31.3 Configuration of the LIN-USART 1300
31.4 LIN-USART pins 1301
31.5 LIN-USART interrupts 1302
31.5.1 Reception interrupt generation and flag set timing 1306
31.5.2 Transmission interrupt generation and flag set timing 1307
31.6 LIN-USART DMA functions 1309
31.7 LIN-USART baud rates 1311
31.7.1 Setting the baud rate 1312
31.7.2 Restarting the reload counter 1314
31.8 Operation of the LIN-USART 1315
31.8.1 Operation in asynchronous mode (operation modes 0 and 1) 1317
31.8.1.1 Operation in asynchronous mode 1317
31.8.2 Operation in synchronous mode (operation mode 2) 1319
31.8.2.1 Operation in synchronous mode (operation mode 2) 1319
31.8.3 Operation of LIN-USART (operation mode 3) 1321
31.8.3.1 Features of LIN-USART in LIN mode (operation mode 3) 1321
31.8.3.2 Operation in asynchronous LIN mode (operation mode 3) 1322
31.8.4 Direct access to serial pins 1323
31.8.5 Bidirectional communication function (normal mode) 1324
31.8.5.1 Bidirectional communication function 1324
31.8.6 Master-slave communication function (multiprocessor mode) 1327
31.8.6.1 Master-slave communication function 1327
31.8.7 LIN communication function 1331
31.8.7.1 LIN master communication function 1331
31.8.8 Sample flowcharts for LIN-USART in LIN communication (operation mode 3) 1333
31.9 Notes on using the LIN-USART 1334

32. 400 kHz I2C Interface 1339

32.1 Overview of I2C Interface 1339
32.2 I2C Interface registers 1341

32.2.1 Bus Control and Status Register (I2Cn_IBCSR) 1342
32.2.2 Ten Bit Slave Address Register (I2Cn_ITBA) 1351
32.2.3 Ten Bit Slave Address Mask Register (I2Cn_ITMK) 1352
32.2.4 Seven Bit Slave Mask and Address Register (I2Cn_ISBMA) 1354
32.2.5 Output Data Register (I2Cn_IODAR) 1356
32.2.6 Clock Control Register (I2Cn_ICCR) 1357
32.2.7 CPU and DMA Input Data Register (I2Cn_ICDIDAR) 1359
32.2.8 Interface Enable and Interrupt Clear Register (I2Cn_IEICR) 1360
32.2.9 Debug and DMA Configuration Register(I2Cn_DDMACFG) 1363
32.2.10 Error Interrupt Enable Register (I2Cn_IEIER) 1365
32.3 Operation of I2C Interface 1368
32.4 Programming flowcharts 1370

33. High Speed SPI Interface 1373

33.1 Outline of the HSSPI 1373
33.2 HSSPI registers 1376
33.2.1 HSSPI Module Control Register (HSSPIn_MCTRL) 1381
33.2.2 HSSPI Peripheral Communication Configuration Register 0~3 (HSSPIn_PCC0~3) 1383
33.2.3 HSSPI TX Interrupt Flag Register (HSSPIn_TXF) 1387
33.2.4 HSSPI TX Interrupt Enable Register (HSSPIn_TXE) 1389
33.2.5 HSSPI TX Interrupt Clear Register (HSSPIn_TXC) 1391
33.2.6 HSSPI RX Interrupt Flag Register (HSSPIn_RXF) 1393
33.2.7 HSSPI RX Interrupt Enable Register (HSSPIn_RXE) 1395
33.2.8 HSSPI RX Interrupt Clear Register (HSSPIn_RXC) 1397
33.2.9 HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF) 1399
33.2.10 HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC) 1401
33.2.11 HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG) 1403
33.2.12 HSSPI Direct Mode DMA Enable Register (HSSPIn_DMDMAEN) 1405
33.2.13 HSSPI Direct Mode Start Register (HSSPIn_DMSTART) 1406
33.2.14 HSSPI Direct Mode Stop Register (HSSPIn_DMSTOP) 1407
33.2.15 HSSPI Direct Mode Peripheral Select Register (HSSPIn_DMPSEL) 1408
33.2.16 HSSPI Direct Mode Transfer Protocol Register (HSSPIn_DMTRP) 1409
33.2.17 HSSPI Direct Mode Byte Count Control Register (HSSPIn_DMBCC) 1410
33.2.18 HSSPI Direct Mode Byte Count Status Register (HSSPIn_DMBCS) 1411
33.2.19 HSSPI Direct Mode Status Register (HSSPIn_DMSTATUS) 1412
33.2.20 HSSPI Transmit Bit Count Register (HSSPIn_TXBITCNT) 1413
33.2.21 HSSPI Receive Bit Count Register (HSSPIn_RXBITCNT) 1414
33.2.22 HSSPI RX Shift Register (HSSPIn_RXSHIFT) 1415
33.2.23 HSSPI TX-FIFO Registers (HSSPIn_TXFIFO0~15) 1416
33.2.24 HSSPI RX-FIFO Registers (HSSPIn_RXFIFO0~15) 1418
33.2.25 HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG) 1420
33.2.26 HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG) 1422
33.2.27 HSSPI Command Sequencer Idle Time Register (HSSPIn_CSITIME) 1424
33.2.28 HSSPI Command Sequencer Address Extension Register (HSSPIn_CSAEXT) 1425
33.2.29 HSSPI Read Command Sequence Data/Control Register 0~7 (HSSPIn_RDCSDC0~7) 1426
33.2.30 HSSPI Write Command Sequence Data/Control Register 0~7 (HSSPIn_WRCSDC0~7) 1428
33.2.31 HSSPI Module ID Register (HSSPIn_MID) 1430
33.3 Operation of the HSSPI 1431
33.3.1 Modes of operation 1431

33.3.2 Retimed clock....1434
33.3.3 Serial clock frequency....1435
33.3.4 SPI protocol....1436
33.3.5 Shift direction....1437
33.3.6 Byte reordering....1443
33.3.7 Safe synchronisation of internal data....1449
33.3.8 Debug mode....1453
33.4 Direct mode....1454
33.5 Command sequencer mode....1460
33.6 Address map of HSSPI....1466
33.7 Notes on using HSSPI....1467

34. Inter IC Sound (I2S) 1477

34.1 Outline of the I2S....1477
34.2 I2S registers....1479
34.2.1 Reception FIFO Data Register (I2Sn_RXFDAT0~15)....1482
34.2.2 Transmission FIFO Data Register (I2Sn_TXFDAT0~15)....1484
34.2.3 Control Register (I2Sn_CNTREG)....1485
34.2.4 Channel Control Register 0 (I2Sn_MCR0REG)....1489
34.2.5 Channel Control Register 1 (I2Sn_MCR1REG)....1493
34.2.6 Channel Control Register 2 (I2Sn_MCR2REG)....1494
34.2.7 Operation Control Register (I2Sn_OPRREG)....1495
34.2.8 Software Reset Register (I2Sn_SRST)....1497
34.2.9 Interrupt Control Register (I2Sn_INTCNT)....1498
34.2.10Status Register (I2Sn_STATUS)....1502
34.2.11DMA Activate Register (I2Sn_DMAACT)....1507
34.2.12Debug Register (I2Sn_DEBUG)....1509
34.2.13Module ID Register (I2Sn_MIDREG)....1510
34.3 I2S frame construction....1511
34.4 I2S configuration and operation modes....1513
34.5 Bit alignment....1519
34.6 FIFO construction....1521
34.7 Caution summary....1523

35. GPIO Module 1525

35.1 Outline of the GPIO module....1525
35.2 GPIO module registers....1527
35.2.1 Data Direction Register (GPIO_DDRi)....1532
35.2.2 Data Direction Set Register (GPIO_DDSRi)....1533
35.2.3 Data Direction Clear Register (GPIO_DDCRi)....1534
35.2.4 Port Output Data Register (GPIO_PODRi)....1535
35.2.5 Port Output Set Register (GPIO_POSRi)....1536
35.2.6 Port Output Clear Register (GPIO_POCRi)....1537
35.2.7 Port Input Data Register (GPIO_PIDRi)....1538
35.2.8 Port Pin Enable Register (GPIO_PPERi)....1539
35.3 Operation of the GPIO module....1541
35.4 GPIO flowchart....1542

36. 8/10-Bit Analog to Digital Converter 1543

36.1 Outline of A/D Converter....1543
36.2 A/D Converter registers....1547
36.2.1 A/D Input Enable Registers (ADCn_ER32, ADCn_ER10)....1556

36.2.2 A/D Control Status Register 0 (ADCn_CS0) 1562
36.2.3 A/D Control Status Register 1 (ADCn_CS1) 1564
36.2.4 A/D Control Status Register 2 (ADCn_CS2) 1567
36.2.5 A/D Control Status Register 3 (ADCn_CS3) 1568
36.2.6 A/D Control Status Register 1 Set Register (ADCn_CSS1) 1570
36.2.7 A/D Control Status Register 1 Clear Register (ADCn_CSC1) 1571
36.2.8 A/D Control Status Register 3 Set Register (ADCn_CSS3) 1572
36.2.9 A/D Control Status Register 3 Clear Register (ADCn_CSC3) 1573
36.2.10 Common Data Register (ADCn_CR) 1574
36.2.11 Dedicated A/D Channel Data Register (ADCn_CD31~0) 1575
36.2.12 ADC Conversion Time Setting Register (ADCn_CT) 1576
36.2.13 A/D Start Channel Setting Register (ADCn_SCH) 1578
36.2.14 A/D End Channel Setting Register (ADCn_ECH) 1579
36.2.15 A/D Converter DMA Configuration Register (ADCn_MAR) 1581
36.2.16 A/D Converter DMA Configuration Set Register (ADCn_MASR) 1582
36.2.17 A/D Converter DMA Configuration Clear Register (ADCn_MACR) 1583
36.2.18 Range Comparator Upper Threshold Registers (ADCn_RCOH3~0) 1584
36.2.19 Range Comparator Lower Threshold Registers (ADCn_RCOL3~0) 1585
36.2.20 A/D Converter Channel Control Registers (ADCn_CC15~0) 1586
36.2.21 Inverted Range Selection Registers (ADCn_RCOIRS32, ADCn_RCOIRS10) 1588
36.2.22 Range Comparator Interrupt Flags (ADCn_RCOINT32, ADCn_RCOINT10) 1600
36.2.23 Range Comparator Over Threshold Flags (ADCn_RCOOF32, ADCn_RCOOF10) 1613
36.2.24 Range Comparator Interrupt Clear Registers (ADCn_RCOINTC32, ADCn_RCOINTC10) 1623
36.2.25 ADC Pulse Counter Reload Registers (ADCn_PCTNRL31~0/ADCn_PCTPRL31~0) 1628
36.2.26 ADC Pulse Counters (ADCn_PCTNCT31~0/ADCn_PCTPCT31~0) 1630
36.2.27 ADC Pulse Counter Zero Flag Registers (ADCn_PCZF32, ADCn_PCZF10) 1632
36.2.28 ADC Pulse Counter Zero Flag Clear Registers (ADCn_PCZFC32, ADCn_PCZFC10) 1640
36.2.29 ADC Pulse Counter Interrupt Enable Registers (ADCn_PCIE32, ADCn_PCIE10) 1646
36.2.30 ADC Pulse Counter Interrupt Enable Set Registers (ADCn_PCIES32, ADCn_PCIES10) 1654
36.2.31 ADC Pulse Counter Interrupt Enable Clear Registers (ADCn_PCIEC32, ADCn_PCIEC10) 1659
36.3 Operation of A/D Converter 1664

37. Bus Support Unit 1673

37.1 Outline of Bus Support Unit 1673
37.2 Bus Support Unit registers 1673
37.2.1 Registers of Bus Support Unit of peripheral 0 group 1673
37.2.1.1 Bus Test Low Register (BSU0_BTSTL) 1676
37.2.1.2 Bus Test High Register (BSU0_BTSTH) 1677
37.2.1.3 Peripheral Enable 0 Low Register (BSU0_PEN0L) 1678
37.2.1.4 Peripheral Enable 1 Low Register (BSU0_PEN1L) 1679
37.2.1.5 Peripheral Enable 2 Low Register (BSU0_PEN2L) 1680
37.2.1.6 Peripheral Enable 3 Low Register (BSU0_PEN3L) 1681
37.2.1.7 Peripheral Enable 4 Low Register (BSU0_PEN4L) 1682
37.2.1.8 Peripheral Enable 5 Low Register (BSU0_PEN5L) 1683
37.2.1.9 Peripheral Enable 6 Low Register (BSU0_PEN6L) 1684
37.2.1.10 Peripheral Enable 7 Low Register (BSU0_PEN7L) 1686

37.2.1.11 Peripheral Enable 7 High Register (BSU0_PEN7H) 1689
37.2.1.12 Peripheral Enable 8 Low Register (BSU0_PEN8L) 1690
37.2.1.13 Peripheral Enable 8 High Register (BSU0_PEN8H) 1691
37.2.1.14 Peripheral Enable 9 Low Register (BSU0_PEN9L) 1692
37.2.1.15 Peripheral Enable 9 High Register (BSU0_PEN9H) 1694
37.2.1.16 Peripheral Enable 11 Low Register (BSU0_PEN11L) 1695
37.2.2 Registers of Bus Support Unit of peripheral 1 group 1696
37.2.2.1 Bus Test Low Register (BSU1_BTSTL) 1698
37.2.2.2 Bus Test High Register (BSU1_BTSTH) 1699
37.2.2.3 Peripheral Enable 0 Low Register (BSU1_PEN0L) 1700
37.2.2.4 Peripheral Enable 1 Low Register (BSU1_PEN1L) 1701
37.2.2.5 Peripheral Enable 2 Low Register (BSU1_PEN2L) 1702
37.2.2.6 Peripheral Enable 3 Low Register (BSU1_PEN3L) 1703
37.2.2.7 Peripheral Enable 4 Low Register (BSU1_PEN4L) 1704
37.2.2.8 Peripheral Enable 5 Low Register (BSU1_PEN5L) 1705
37.2.2.9 Peripheral Enable 6 Low Register (BSU1_PEN6L) 1706
37.2.2.10 Peripheral Enable 7 Low Register (BSU1_PEN7L) 1707
37.2.2.11 Peripheral Enable 8 Low Register (BSU1_PEN8L) 1708
37.2.2.12 Peripheral Enable 9 Low Register (BSU1_PEN9L) 1709
37.2.2.13 Peripheral Enable 9 High Register (BSU1_PEN9H) 1711
37.2.2.14 Peripheral Enable 10 Low Register (BSU1_PEN10L) 1712
37.2.2.15 Peripheral Enable 10 High Register (BSU1_PEN10H) 1713
37.2.2.16 Peripheral Enable 11 Low Register (BSU1_PEN11L) 1714
37.2.2.17 Peripheral Enable 11 High Register (BSU1_PEN11H) 1715
37.2.3 Registers of Bus Support Unit of peripheral 3 group 1716
37.2.3.1 Bus Test Register (BSU3_BTST) 1717
37.2.3.2 Peripheral Enable 2 Register (BSU3_PEN2) 1718
37.2.3.3 Peripheral Enable 4 Register (BSU3_PEN4) 1720
37.2.4 Registers of Bus Support Unit of peripheral 4 slave group 1721
37.2.4.1 Bus Test Register (BSU4_BTST) 1722
37.2.4.2 Peripheral Enable 1 Register (BSU4_PEN1) 1723
37.2.4.3 Peripheral Enable 2 Register (BSU4_PEN2) 1724
37.2.4.4 Peripheral Enable 3 Register (BSU4_PEN3) 1725
37.2.4.5 Peripheral Enable 4 Register (BSU4_PEN4) 1726
37.2.4.6 Peripheral Enable 5 Register (BSU4_PEN5) 1727
37.2.4.7 Peripheral Enable 6 Register (BSU4_PEN6) 1728
37.2.4.8 Peripheral Enable 7 Register (BSU4_PEN7) 1729
37.2.4.9 Peripheral Enable 8 Register (BSU4_PEN8) 1730
37.2.5 Registers of Bus Support Unit of peripheral 5 group 1731
37.2.5.1 Bus Test Register (BSU5_BTST) 1732
37.2.6 Registers of Bus Support Unit of MEMORY_CONFIG group 1733
37.2.6.1 Bus Test Register (BSU6_BTST) 1734
37.2.6.2 Peripheral Enable 2 Register (BSU6_PEN2) 1735
37.2.6.3 Peripheral Enable 4 Register (BSU6_PEN4) 1736
37.2.6.4 Peripheral Enable 12 Register (BSU6_PEN12) 1737
37.2.6.5 Peripheral Enable 20 Register (BSU6_PEN20) 1738
37.2.7 Registers of Bus Support Unit of the MCU_CONFIG group 1739
37.2.7.1 Bus Test Register (BSU7_BTST) 1740
37.2.7.2 Peripheral Enable 3 Register (BSU7_PEN3) 1741
37.2.7.3 Peripheral Enable 5 Register (BSU7_PEN5) 1742
37.2.7.4 Peripheral Enable 7 Register (BSU7_PEN7) 1743
37.2.7.5 Peripheral Enable 8 Register (BSU7_PEN8) 1745
37.2.8 Registers of Bus Support Unit of the HSSPI0 group 1747

37.2.8.1 Bus Test Register (BSU8_BTST) 1748
37.2.8.2 Peripheral Enable 0 Register (BSU8_PEN0) 1749
37.2.9 Registers of Bus Support Unit of the EBI group 1750
37.2.9.1 Bus Test Register (BSU10_BTST) 1751
37.2.9.2 Peripheral Enable 0 Register (BSU10_PEN0) 1752
37.3 Operation of Bus Support Unit 1753

38. Up/Down Counter 1755

38.1 Outline of Up/Down Counter (UDC) 1755
38.2 Up/Down Counter registers 1760
38.2.1 Count Control Register 0 (UDCn_CC0) 1761
38.2.2 Count Control Register 1 (UDCn_CC1) 1763
38.2.3 Extended Count Control Register 0 (UDCn_ECC0) 1765
38.2.4 Extended Count Control Register 1 (UDCn_ECC1) 1767
38.2.5 Count Status Register 0 (UDCn_CS0) 1769
38.2.6 Count Status Register 1 (UDCn_CS1) 1771
38.2.7 Up/Down Counter Register (UDCn_CR) 1773
38.2.8 Up/Down Reload/Compare Register (UDCn_RC) 1774
38.2.9 Count Toggle Register 0 (UDCn_TGL0) 1776
38.2.10 Count Toggle Register 1 (UDCn_TGL1) 1778
38.2.11 Count Debug Register (UDCn_DBG) 1780
38.3 Operation of Up/Down Counter 1781
38.3.1 Procedure for configuring the UDC 1793
38.3.1.1 Selection of bit length (32 or 16) of the UDC 1793
38.3.1.2 Number of count modes available and how are they set 1793
38.3.1.3 Selection of count source for UDC running in the timer mode 1793
38.3.1.4 Selection of edge with which UDC running in the up/down count mode detects an input signal (AIN or BIN) 1794
38.3.1.5 Setting a value to the UDC 1794
38.3.1.6 How to enable clearing of the UDC the next time the counter counts up after the UDC's count-up value matches the compare value (UDCn_RC) 1794
38.3.1.7 Method to reload value (UDCn_RC) to UDC when it has underflowed 1794
38.3.1.8 Method to clear UDC 1794
38.3.1.9 Method to clear UDC using the ZIN pin 1795
38.3.1.10 Method to control UDC's count operation using the ZIN pin 1795
38.3.1.11 Method to enable/disable UDC's count operation 1796
38.3.1.12 Method to know the previous count direction (the current rotation direction) 1796
38.3.1.13 Method to know count direction changes 1796
38.3.1.14 Method to know that a compare-match has occurred 1797
38.3.1.15 Method to know that an overflow or underflow has occurred 1797
38.3.1.16 Method to set the Reload/Compare value 1797
38.3.1.17 Method to configure interrupt-related registers 1797
38.3.1.18 Number of interrupts available and how they are selected 1797
38.3.1.19 Method to enable (select), disable, or clear interrupts 1798
38.3.1.20 Method to configure toggle output for change direction events 1798
38.3.1.21 Method to clear toggle output 1798
38.3.1.22 Method to stop counters in debug mode 1798

39. Automotive Remote Handler 1801

39.1 Outline of ARH 1801

39.2 ARH registers 1805
39.2.1 ARH Remote Handler Control Register (ARHn_RHCTRL) 1811
39.2.2 ARH Channel Control Register (ARHn_CHCTRL0 - Channel 0) 1814
39.2.3 ARH Channel Control Register (ARHn_CHCTRL1 - Channel 1) 1820
39.2.4 ARH Channel Status Register (ARHn_CHSTAT0 - Channel 0, ARHn_CHSTAT1 - Channel 1) 1825
39.2.5 ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0 - Channel 0, ARHn_CHWDGCTL1 - Channel 1) 1826
39.2.6 ARH Watchdog Counter Register (ARHn_CHWDGCNT0 - Channel 0, ARHn_CHWDGCNT1 - Channel 1) 1829
39.2.7 ARH Transaction Buffer Control Register (ARHn_TBCTRL0~15) 1830
39.2.8 ARH Transaction Buffer Interrupt Register (ARHn_TBIRQ) 1834
39.2.9 ARH Transaction Buffer Index Register (ARHn_TBIDX0 - Channel 0, ARHn_TBIDX1 - Channel 1) 1835
39.2.10 ARH Transaction Frame Control Register (ARHn_TFCTRL00~15) 1836
39.2.11 ARH Transaction Frame Index Register (ARHn_TFIDX0~15) 1840
39.2.12 ARH Transaction Frame Address Register (ARHn_TFADDR0~15) 1841
39.2.13 ARH Transaction Frame Data Register (ARHn_TFDATA0~15) 1842
39.2.14 ARH Event Control Register (ARHn_EVCTRL) 1844
39.2.15 ARH Event Interrupt Control Register (ARHn_EVIRQC) 1845
39.2.16 ARH Event Buffer Register (ARHn_EVBUF0) 1847
39.2.17 ARH Event Buffer Register (ARHn_EVBUF1) 1848
39.2.18 ARH APIX® Configuration Register (ARHn_APCFG00 - Channel 0) 1849
39.2.19 ARH APIX® Configuration Register (ARHn_APCFG10 - Channel 1) 1853
39.2.20 ARH APIXR Configuration Register (ARHn_APCFG01 - Channel 0) 1856
39.2.21 ARH APIX® Configuration Register (ARHn_APCFG11 - Channel 1) 1858
39.2.22 ARH APIX® Configuration Register (ARHn_APCFG02 - Channel 0) 1860
39.2.23 ARH APIX® Configuration Register (ARHn_APCFG12 - Channel 1) 1863
39.2.24 ARH APIX® Configuration Register (ARHn_APCFG03 - Channel 0, ARHn_APCFG13 - Channel 1) 1865
39.2.25 ARH Test Register (ARHn_TST) 1868
39.2.26 ARH Unlock Register (ARHn_UNLOCK) 1869
39.2.27 ARH Module ID Register (ARHn_MID) 1870
39.2.28 ARH APIX® Configuration Register (ARHn_APCFG04 - Channel 0, ARHn_APCFG14 - Channel 1) 1871
39.2.29 APIX® PHY Evaluation Transmitter Pattern Register (ARHn_EVAL0) 1873
39.2.30 APIX® PHY Evaluation Receiver Pattern Register (ARHn_EVAL1) 1874
39.2.31 APIX® PHY Evaluation Configuration Register (ARHn_EVAL2) 1875
39.2.32 APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3) 1878
39.2.33 APIX® PHY Evaluation Misc Test Enable Register (ARHn_EVAL4) 1880
39.2.34 APIX® PHY Evaluation Status Register (ARHn_EVAL5) 1881
39.2.35 APIX® PHY Evaluation Misc Test Configuration Register (ARHn_EVAL6) 1882
39.2.36 APIX® PHY Evaluation Misc Test Status Register (ARHn_EVAL7) 1883
39.3 Operation of ARH 1884
39.3.1 Data transfers (frame types) 1884
39.4 Notes on using ARH 1892
39.5 Use cases 1893
39.5.1 Communication over Automotive Interconnect to external A-Shell 1893
39.5.2 Communication over Automotive Interconnect to external PHY 1895
39.5.3 Communication over APIX® link (internal PHY) to Remote RX PHY 1896
39.5.4 Communication over Automotive Interconnect-2 to External Automotive Interconnect-2 1897

40. High-Performance Matrix 1899
40.1 Outline of the High-Performance Matrix ... 1899
40.2 Arbitration scheme for the HPM ... 1901

41. DMA Controller 1903
41.1 Overview of the DMA Controller ... 1903
41.2 DMA Controller registers ... 1905
41.2.1 DMA Controller Global Configuration Register (DMAn_R) ... 1908
41.2.2 DMA Controller Global Completion Interrupt 1 Register (DMAn_DIRQ1) ... 1911
41.2.3 DMA Controller Global Completion Interrupt 2 Register (DMAn_DIRQ2) ... 1912
41.2.4 DMA Controller Global Error Interrupt 1 Register (DMAn_EDIRQ1) ... 1913
41.2.5 DMA Controller Global Error Interrupt 2 Register (DMAn_EDIRQ2) ... 1914
41.2.6 DMA Controller ID Register (DMAn_ID) ... 1915
41.2.7 DMA Controller Channel Configuration A Register Channel 'i' (DMAn_Ai) ... 1916
41.2.8 DMA Controller Channel Configuration B Register Channel 'i' (DMAn_Bi) ... 1920
41.2.9 DMA Controller Channel Configuration Source Address Register Channel 'i' (DMAn_SAi) ... 1924
41.2.10 DMA Controller Channel Configuration Destination Address Register Channel 'i' (DMAn_DAi) ... 1925
41.2.11 DMA Controller Channel Configuration C Register Channel 'i' (DMAn_Ci) ... 1926
41.2.12 DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di) ... 1927
41.2.13 DMA Controller Channel Configuration Source Address Shadow Register Channel 'i' (DMAn_SASHDWi) ... 1930
41.2.14 DMA Controller Channel Configuration Destination Address Shadow Register Channel 'i' (DMAn_DASHDWi) ... 1931
41.2.15 DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk) ... 1932
41.2.16 DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj) ... 1936
41.2.17 DMA Controller Client Matrix Channel Interface Configuration Register 'i' (DMAn_CMCHICi) ... 1938
41.3 Operation of the DMA Controller ... 1939
41.3.1 DMA channels ... 1941
41.3.1.1 Modes of operation ... 1941
41.3.1.2 Block transfer mode ... 1941
41.3.1.3 Burst transfer mode ... 1948
41.3.1.4 Demand transfer mode ... 1948
41.3.2 DMA Client Matrix ... 1949
41.3.2.1 Overview ... 1949
41.3.2.2 Modes of operation ... 1950
41.3.2.3 Functional description ... 1950
41.3.2.4 Initialization and application information ... 1951
41.3.3 DMA arbiter ... 1952
41.3.3.1 Overview ... 1952
41.3.3.2 Fixed priority ... 1952
41.3.3.3 Dynamic priority ... 1953
41.3.3.4 Round-robin ... 1954
41.3.3.5 Application information ... 1955
41.3.4 DMA AHB slave interface ... 1955
41.3.4.1 Supported data transfers ... 1956
41.3.4.2 Data transfer response ... 1956
41.3.5 External DMA Client Interface signal requirements ... 1956

42. External Bus Interface 1961

42.1 Outline of the External Bus Interface 1961
42.1.1 Pad interface 1962
42.1.2 External Bus Interface and Memory regions 1963
42.2 External Bus Interface Configuration registers 1964
42.2.1 Unlock Register (EBI_UNLOCK) 1967
42.2.2 Lock Status Register (EBI_LSTSR) 1968
42.2.3 SRAM/FLASH Mode Control Register (EBI_SFMR0~7) 1969
42.2.4 SRAM/FLASH Access Configuration Register (EBI_SFACCR0~7) 1972
42.2.5 SRAM/FLASH Address Control Register (EBI_SFADDCR0~7) 1975
42.2.6 SDRAM Mode Control Register (EBI_SDMODCR) 1977
42.2.7 SDRAM Refresh Control Register (EBI_SDRCR) 1982
42.2.8 SDRAM Power Control Register (EBI_SDPCR) 1984
42.2.9 SDRAM Timing Control Register (EBI_SDTCR) 1985
42.2.10 SDRAM Command Register (EBI_SDCOMDR) 1987
42.2.11 Error Register (EBI_ERRR) 1989
42.3 Operation of the External Bus Interface 1990
42.3.1 SRAM/FLASH controller 1990
42.3.2 SDRAM controller 1991
42.3.3 NAND FLASH access 1992
42.3.4 Endianess and enabled byte lanes 1995
42.4 External memory connection to the External Bus Interface 1998
42.5 Access waveform examples for the External Bus Interface 2002

43. Secure Hardware Extension (SHE) 2007

43.1 Outline of the Secure Hardware Extension (SHE) module 2007
43.2 Secure Hardware Extension registers 2009
43.2.1 Configuration registers for the Command Interface 2015
43.2.1.1 SHE Command Register (SHE_CMD) 2015
43.2.1.2 SHE Command Cancel Register (SHE_CMDCANCEL) 2017
43.2.1.3 SHE Clock Control Register (SHE_CLKCTRL) 2018
43.2.2 Status registers for the command interface 2020
43.2.2.1 SHE Status Register (SHE_STATUS) 2020
43.2.2.2 SHE Error Code Register (SHE_ERC) 2024
43.2.2.3 SHE Clock Status Register (SHE_CLKSTAT) 2027
43.2.2.4 SHE Module ID Register (SHE_MID) 2028
43.2.3 Interrupt registers for command interface 2029
43.2.3.1 Interrupt Request Register (SHE_IRQ) 2029
43.2.3.2 Interrupt Request Enable Register (SHE_IRQEN) 2032
43.2.3.3 Interrupt Request Clear Register (SHE_IRQCLR) 2034
43.2.4 Configuration registers for the data interface 2036
43.2.4.1 Input Channel Master Start Address Register (SHE_IMSTADDR) 2036
43.2.4.2 Output Channel Master Start Address Register (SHE_OMSTADDR) 2037
43.2.4.3 Input Channel Master Data Transfer Counter (SHE_IMSTCNT) 2038
43.2.4.4 Output Channel Master Data Transfer Counter (SHE_OMSTCNT) 2039
43.2.4.5 Input Channel Master Start Trigger (SHE_IMSTSTART) 2040
43.2.4.6 Output Channel Master Start Trigger (SHE_OMSTSTART) 2041
43.2.4.7 Input FIFO Configuration Register (SHE_IFIFOCFG) 2042
43.2.4.8 Output FIFO Configuration Register (SHE_OFIFOCFG) 2044
43.2.4.9 Input FIFO Compare Value Register (SHE_COMPARE0~1) 2046
43.2.5 Status registers for data interface 2048
43.2.5.1 Compare Register Access Status Register (SHE_COMPACC) 2048

43.2.5.2 Data Master Status Register (SHE_MSTSTATUS) 2049
43.2.5.3 Input Channel Master Error Response Address Register (SHE_IMSTERRADDR) 2053
43.2.5.4 Output Channel Master Error Response Address Register (SHE_OMSTERRADDR) 2054
43.2.5.5 FIFO Status Register (SHE_FIFOSTATUS) 2055
43.2.5.6 FIFO Load Register (SHE_FIFOLOAD) 2057
43.2.5.7 Input FIFO Data Counter Register (SHE_DATACNT0~1) 2058
43.2.6 Data transfer registers 2060
43.2.6.1 Input FIFO Write Data Register (SHE_IFIFOWRDATA0~31) 2060
43.2.6.2 Output FIFO Read Data Register (SHE_OFIFORDDATA0~31) 2061
43.3 Operation of the Secure Hardware Extension 2062
43.3.1 Operation of the command interface 2062
43.3.1.1 Power saving functionality 2071
43.3.1.2 Command interface access issues 2072
43.3.2 Operation of the data interface 2073
43.3.3 Operation of the FIFO unit 2077
43.4 Notes on using SHE 2080
43.4.1 General notes on using SHE 2080
43.4.1.1 Notes on using SHE commands 2080
43.4.1.2 SHE Used Error Codes/Used Cancel Codes 2081
43.5 References 2085

44. Media Local Bus Interface (MediaLB) 2087

44.1 Overview of MediaLB 2087
44.2 MediaLB Register Set 2094
44.2.1 MediaLBn Device Control Configuration Register (MLBn_DCCR) 2100
44.2.2 MediaLBn System Status Configuration Register (MLBn_SSCR) 2103
44.2.3 MediaLBn System Data Configuration Register (MLBn_SDCR) 2106
44.2.4 MediaLBn System Mask Configuration Register (MLBn_SMCR) 2107
44.2.5 MediaLBn Version Control Configuration Register (MLBn_VCCR) 2110
44.2.6 MediaLBn Synchronous Base Address Configuration Register (MLBn_SBCR) 2111
44.2.7 MediaLBn Asynchronous Base Address Configuration Register (MLBn_ABCR) 2112
44.2.8 MediaLBn Control Base Address Configuration Register (MLBn_CBCR) 2113
44.2.9 MediaLBn Isochronous Base Address Configuration Register (MLBn_IBCR) 2114
44.2.10 MediaLBn Channel Interrupt Configuration Register (MLBn_CICR) 2115
44.2.11 MediaLBn AHB Master Control Register (MLBn_AHBMCTL) 2116
44.2.12 MediaLBn Channel Entry Configuration Register (MLBn_CECR0 - MLBn_CECR15) 2117
44.2.13 MediaLBn Channel Status Configuration Register (MLBn_CSCR0 - MLBn_CSCR15) 2121
44.2.14 MediaLBn Channel Current Buffer Configuration Register (MLBn_CCBCR0 - MLBn_CCBCR15) 2128
44.2.15 MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0 - MLBn_CNBCR15) 2130
44.2.16 MediaLBn Local Channel Buffer Configuration Register (MLBn_LCBCR0 - MLBn_LCBCR15) 2133
44.2.17 MediaLBn Module Identification Register (MLBn_MID) 2135
44.3 Notes on Using MediaLB 2136
44.4 List of References 2137

Revision History 2139

1. Overview

Revision history

Revision Number	Date	Modification
0.01	15-Oct- 2009	Preliminary draft
0.02	25-Nov-2009	Updated four modules (Real Time Clock, BOOTROM Hardware Interface, TCM RAM Interface, Timing Protection Unit modules) in 'Draft 2' format as per requirement.
0.03	30-Nov-2009	■ Compiled FRT, OCU and ICU into one module (16-bit I/O Timer). ■ Updated the manual for two modules (16-bit I/O Timer, 32-bit Reload Timer) in 'Draft 2' format.
0.04	7-Dec-2009	■ Updated the manual for four modules (Programmable CRC, External Interrupt Capture Unit, Error Collection Unit and AHB RAM Interface) in 'Draft 2' format. ■ Increased the spacing between the footer and the text throughout the manual. ■ Updated bookmark section.
0.05	11-Dec-2009	■ Updated the manual for three modules (Programmable Pulse Generator, Stepper Motor Controller, Up/Down Counter) in 'Draft 2' format.
0.06	21-Dec-2009	■ Updated the manual for five modules (Sound Generator, 400 kHz I2C Interface, High Speed SPI Interface, Inter IC Sound and GPIO Module) in 'Draft 2' format.
0.07	11-Jan-2010	■ Updated the completed modules for review comments ■ Included four modules (Main Flash, Port Mux, External Interrupts, SRAM) in Draft 2 format. ■ Included four modules (MPU, Interrupt Controller, PPU, and CAN) in 'Draft 1' format ■ Updated the Overview section for legends and abbreviations
0.08	18-Jan-2010	■ Updated the manual for four modules in Draft 2 format (MPU, PPU, Interrupt Controller and ADC). ■ Updated four modules for register name changes (I2C, I2S, HSSPI, TPU)
0.09	10-Feb-2010	■ Updated the manual for two modules in Draft 2 format (Security Checker, EEPROM Emulation Flash). ■ Updated the manual for four modules in Draft 1 format (System Controller, LIN-USART, CAN Controller, 10/100 MBPS Ethernet MAC Controller, Automotive Remote Handler) ■ Updated two modules for register name changes (MPU and Interrupt Controller)

Revision Number	Date	Modification
0.10	26-Feb-2010	■ Updated PPU module for register name changes ■ Added Chapter 39 HPM in Draft 1 format (High Performance Matrix) ■ Updated six modules in Draft 3 format (CRC, TCMRAM, BECU, SMC, GPIO, and UDC) ■ Updated Overview chapter for abbreviations ■ Corrected the header section to include Chapter name with title on all the modules ■ Changed 'Read 0' to 'read 0' for modules in Draft 3 format
0.11	17-Mar-2010	■ Updated three modules in Draft 1 format (System Controller, Remote Handler, Ethernet MAC) ■ Updated one module in Draft 2 format (LIN USART) ■ Updated seven modules in Draft 3 format (EICU, ICU, OCU, I2C, I2S, BootROM, TPU)
0.11	25-Mar-2010	■ Reverted back a few modules to Draft 2 format (ICU, OCU, I2C, I2S, BootROM, TPU, SMC, TCMRAM) as bugs were not closed
0.12	12-Apr-2010	■ Updated the overview section for abbreviations ■ Updated TCMRAM, SRAM_IF, GPIO, External Interrupts for bugzilla ■ Updated register name changes for MPU, Main Flash and RTC module ■ Reverted back a few modules to Draft 1 format as per the new schedule (Security Checker, Retention RAM, LIN USART)
0.13	14-Jun-2010	■ Reworked on all the modules and updated the legends ■ Updated one module GPIO in Draft 5 format (GPIO) ■ Updated six modules in Draft 4 format (Watchdog, CRC, SRAM, EICU, BECU, UDC) ■ Updated seven modules in Draft 3 format (BootROM, Sound Generator, I2C, I2S, TCMRAM, RLT, and SMC) ■ Updated 14 modules in Draft 2 format (RTC, Security Checker, MPU AXI, MPU AHB, TCFLASH, EEPROM Flash, Interrupt Controller, Port Mux, TPU, PPU, External Interrupts, ■ 16-bit I/O Timer, PPG, and HSSPI) ■ Updated 9 modules in Draft 1 format (Overview, System Controller, AHB RAM, CAN Controller, LIN USART, ADC, Ethernet MAC, Automotive Remote Handler, High Performance Matrix) ■ Dropped Security Bridge chapter and replaced it with MPU AXI chapter
0.14	25-Jun-2010	■ Updated 1 module in Draft 3 format (TCFLASH) ■ Updated 3 modules in Draft 2 format (Retention RAM, Ethernet MAC, and PPG) ■ Corrected Security Checker module in Draft 2 format
0.15	19-July-2010	■ Corrected problem with corrupted files released in version 0.14 ■ Fixed bugs for EEPROM module (bug # 3137) and RTC module (bug # 2124)

Revision Number	Date	Modification
0.16	16-Nov-2010	■ Intermediate release, included front page and back page ■ Changed the watermark for Draft 4 and above to 'PRELIMINARY'. Updated 16 modules in 'PRELIMINARY' format (RTC, Watchdog, CRC, TCFLASH, EEPROM FLASH, TCMRAM, SRAM, EICU, BECU, SGE, PPG, RLT, I2C, I2S, GPIO, UDC) ■ Updated 7 modules in 'DRAFT 3' format (SMC, BOOTROM, SECURITY CHECKER, PORT MUX, AHBRAM, EXT INT, ETHERNET(■ Updated 11 modules in 'DRAFT 2' format (SYSTEM CONTROLLER, MPU AXI, PPU, TPU, IUNIT, IOTIMER, LIN, ADC, HSSPI, BSU, ARH) ■ Updated 3 modules in 'DRAFT 1' format (RESOURCE MUX, CAN, OVERVIEW) ■ Changed the watermark to 'OBSOLETE' for modules to be removed or replaced on a later date (MPU AHB, VRAM, and HIGH PEROFORMANCE MATRIX) ■ Fixed all the modules for consistent writing style for binary numbers, hexadecimal numbers, and for consistency ■ Fixed all the modules for bug 5104 ■ Updated the series number to FCR4 Cluster series
0.17	-	■ Skipped to keep consistency with all chapter numbering
0.18	13-Jun-2011	■ New book release to reflect status of chapters that have changed ■ Overview and Security Checker chapters are now draft-3 ■ RTC, Watchdog, EICU, External Interrupts, Sound Generator, PPG, RLT, SMC, GPIO, and Ethernet MAC chapters are preliminary
0.19	15-Aug-2011	■ Updated CRC and I2C chapters
1.0	26-Sept-2012	■ New book release with updated chapters except for the AHB_MPU, I/O Timer, HPM, DMAC and RICFG chapters, which feature no modifications since revision number 0.19. These chapters are marked as "preliminary". Other chapters marked as preliminary are those which still require some small modification. ■ The HPM chapter in revision number 0.19 is obsolete and will be rewritten. ■ Additional EEFLASH chapter (Chapter 10A) for devices with SHE; additional Boot ROM software interface chapter (Chapter 14A); and addition of EBI chapter (Chapter 40); SHE chapter (Chapter 41); and MediaLB chapter (Chapter 42). ■ Ethernet chapter (Chapter 36 in revision number 0.19) has been removed.
1.1	09-Nov-2012	■ New book release with updated AHB_MPU, I/O Timer, HPM, DMAC and RICFG chapters. All known corrections implemented.
1.2	07-June-2013	■ Known inconsistencies from version 1.1 fixed. ■ Chapter 14A - Table 14A.2-1 extended to include information for MB9DF125 and MB9E226. ■ Chapter 41 (SHE) - Addition of used error codes/ used cancel codes.

1.1 Overview

This section describes the features and block diagram of the FCR4 Cluster series.

1.1.1 Features of the FCR4 Cluster series

Table 1-1. Features overview table

Features	FCR4 Cluster series
Core clock frequency	up to 160 MHz
Instruction cache	up to 16 KB
Data cache	up to 16 KB
Tightly Coupled Flash (TCFLASH)	up to 2 MB + 128 KB
EEPROM Emulation Flash (EEFLASH)	up to 64 KB
Tightly Coupled Memory (TCMRAM)	up to 128 KB
System RAM	up to 64 KB
Retention RAM	up to16 KB
Video RAM	external SDRAM or up to 2MB internal SRAM
Trace data width	up to 32 bit
Graphics resolution	1024 pixels x 1024 pixels
DMA	up to 8 channels
External Interrupts (EIC)	up to 32 channels
External Non-Maskable Interrupts	up to 1 channel
CAN	up to 3 channels
I2C	up to 1 channel
USART/LIN	up to 2 channels
A/D Converters (ADC)	up to 26 + 6 * 4 channels
Stepper Motor Control (SMC)	up to 6 channels
Ethernet (ETH)	up to 1 channel
SPI	up to 3 channels
16-bit Free Running Timer (FRT)	up to 8 channels
16-bit Input Capture (ICU)	up to 8 channels
16-bit Output Compare (OCU)	up to 8 channels
Reload Timer (RLT)	up to 10 channels

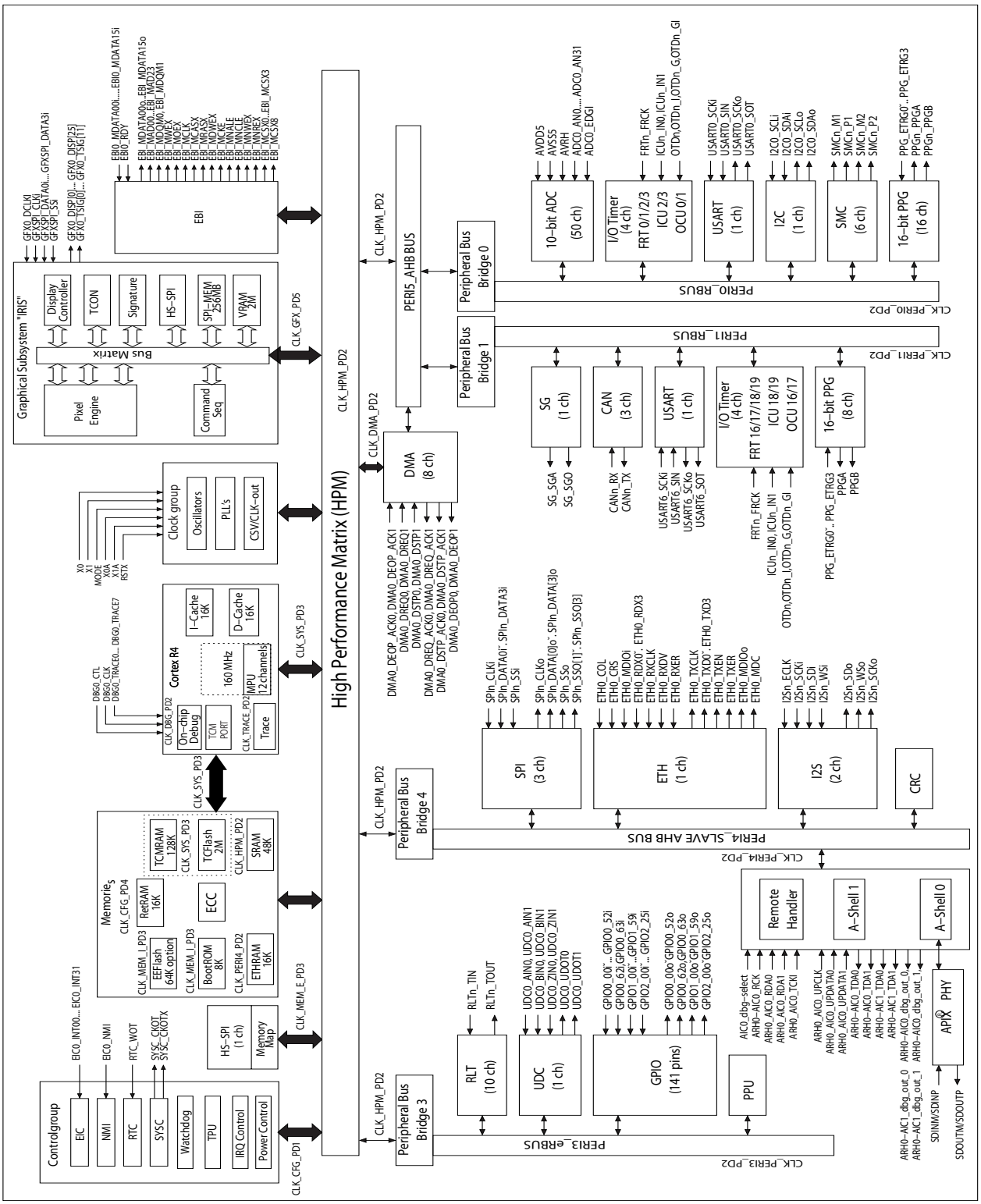
Table 1-1. Features overview table

Features	FCR4 Cluster series
16-bit Programmable Pulse Generator (PPG)	up to 24 16-bit channels / 48 8-bit channels
Sound Generator (SG)	up to 1 channel
Inter IC Sound (I2S)	up to 2 channels
Up Down Counter (UDC)	up to 1 32-bit channel / 2 16-bit channels
Quad SPI	up to 2 channels
Real Time Clock (RTC)	up to 1 unit
I/O ports	up to 141 pins
Clock Supervisors	up to 5 units
Watchdog Timer (WDG)	1 unit
Low Voltage Detect	up to 3 units
Timing Protection Unit (TPU)	up to 1 unit (8 channels)
APIX	up to 2 channels
External Bus	up to 24-bit address/16-bit data
Media LB (future extension)	up to 1 channel (3 wire)

Note: For the exact features available in a device refer to the device-specific datasheet.

FCR4 Cluster series block diagram

The FCR4 Cluster series comprises of four devices which includes Calypso, Atlas, Atlas-L and Titan.



1.2 General notes on using this document

This section contains general notes about this document.

Device dependent resources

The derivatives of a Microcontroller series may have different sets of resources (e.g. features, number of channels, interconnections). Details about an individual device can be found in the datasheet of the device. These details include:

- The number of resources and available channels
- The peculiarities of a resource in a derivative (e.g. interconnections between resources)

Described resources

Chapters concerned with resources of the Microcontroller, such as USART, Reload Timers and General Purpose Ports describe the function of only one channel of this resource. The behavior of all other resources of the same type is the same. Any exceptions are described in this chapter or in the corresponding chapter of this hardware manual.

In the hardware manual chapters, a placeholder, usually denoted by ‘n’ is used for a resource channel number (e.g. in the register name). This placeholder needs to be replaced with the resource channel number of the resource to be used.

Example: Reload Timer Registers

Register name as in the Chapter ‘32-bit Reload Timer’	Register name define in header file for Reload Timer 0	Register name defined in header file for Reload Timer 1
RLTn_DMACFG	RLT0_DMACFG	RLT1_DMACFG
RLTn_TMCSR	RLT0_TMCSR	RLT1_TMCSR
RLTn_TMRLR	RLT0_TMRLR	RLT1_TMRLR
RLTn_TMR	RLT0_TMR	RLT1_TMR

1.3 Legend

This section describes the legend used throughout this manual.

The following legend applies to the bit mode row in the register description tables in the subsequent sub-sections:

Table 1-2. Legend

Symbols	Meaning
'-':	Access attribute for a reserved bit
X:	Read access returns an unknown value
R:	Read
Rd	Read if device security is off. Otherwise reading returns an error response
Rf:	Read/execute. Sector-wise configurable by Flash security permission set
Rl:	Read/execute. Configurable by Link Key
RP:	Read in privileged mode. Reading in non-privileged mode causes an error response.
Rp:	Read in privileged mode, configurable. Reading in non- privileged mode causes an error response unless the corresponding access or read attributes are set in the PPU
R(P,p)0:	Read always 0 (privileged, privileged config (PPU))
R(P,p)1:	Read always 1 (privileged, privileged config (PPU))
R(P,p)X:	Read value is not determinable (privileged, privileged config (PPU))
W:	Write
Wf	Write/erase. Sector-wise configurable by Flash security permission set
WP:	Write in privileged mode. Writing in non-privileged mode causes error response.
Wp:	Write in privileged mode, configurable. Writing in non-privileged mode causes an error response unless the corresponding access attribute is set in the PPU.
W(P,p,S) 0:	Write 0 (privileged, privileged config (PPU), with protected sequence). Writing 1 has no effect.
W(P,p,S) 1:	Write 1 (privileged, privileged config (PPU), with protected sequence). Writing 0 has no effect.

Note: A combination of symbol for writes and symbol for reads is permitted. In such cases, a read symbol precedes a write symbol.

1.3.1 Global description of register table

[REG]										
31	**30**	**29**	**28**	**27**	**.....Bit position.....(in decimal)**	**04**	**03**	**02**	**01**	**00**
					.....Bit name.....					
					.....Bit mode.....					
					.....Reset/default value.....					

1.4 Abbreviations

This section lists the terms and abbreviations used in this manual.

Table 1-3. Terms and abbreviations

Term	Meaning
AHB	Advanced High-Performance Bus
AIC	Automotive Interconnect
AMBA™	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
APIX ®	Automotive Pixel Channel
AR	Auto Reply
ARH	Automotive Remote Handler
AShell	Automotive Shell
ATCM	TCM-A port of Arm® Cortex® R4 CPU
AXI	Advanced eXtensible Interface
BADDR	Buffer Address
BD	Buffer Descriptor
BDC	Buffer Descriptor Controller
BDR	Boot Description Record
BDT	Buffer Descriptor Table
BECU	Bus Error Collection Unit
BSP	Bit Stream Processor
BSU	Bus Support Unit
BTL	Bit Timing Logic
Byte Time	Byte time is the time required for transmission of a byte of data, over the SPI interface. One byte-time is 2 cycles of SCLK in quad-bit SPI protocol, 4 cycles of SCLK in dual-bit SPI protocol and 8 cycles of SCLK in legacy SPI protocol.
CAN	Controller Area Network
CBC	Cipher-Block Chaining
CPU	Central Processing Unit
CMAC	Cipher-based Message Authentication Code
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access / Collision Detect
CSR	Configuration and Status Register, Control and Status Registers
CSV	Clock Supervisor
DAP	Debug Access Port
DED	Double Error Detection
DLC	Data Length Code
DMA	Direct Memory Access
DMAACK	DMA Request Acknowledgement
DMAREQ	DMA Request
ECC	Error Correction Code

Table 1-3. Terms and abbreviations

Term	Meaning
EEFLASH	EEPROM Emulation Flash
EML	Error Management Logic
ETHERNET	10/100 ETHERNET module
ETHERNETRAM	RAM Interface module instantiated in ETHERNET
EVBUF	Event Buffer
FastMAC	Fast-Ethernet MAC module instantiated in ETHERNET
FCS	Frame Check Sequence
FF	Flexi Filters in FastMAC
FFA	Field Failure Analysis
FIFO	First-In, First-Out
FIQ	Fast Interrupt Request
FPP	Flash Parallel Programming
FRT	Free Running Timer
FSM	Finite State Machine
GFX	Graphics
GPIO	General Purpose Input/Output
Half Word	16 bits of data
HP	High Priority
HPM	High Performance Matrix
HRESP	Response from AHB slave
HREADY	Ready signal from AHB slave
HRDATA	Read data bus from AHB slave
I/F	Interface
ICU	Input Capture Unit
IF	Interface
IFS	Inter-Frame Spacing
IR	Interrupt
IRQ	Interrupt Request
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
LRG	Least Recently Granted
LSB	Least Significant Bit
LVD	Low Voltage Detector
MAC	Media Access Control
MCU	Microcontroller Unit
MDIO	Management Data Input / Output
MII	Media Independent Interface
MPM	Magic Packet Mode
MPU	Memory Protection Unit
MPU AHB	Memory Protection Unit for AHB memory space

Table 1-3. Terms and abbreviations

Term	Meaning
NF	Noise Filter
NGM	Next Generation MCU
NMI	Non-Maskable Interrupt
OCU	Output Compare Unit
OSC	Oscillator
PA	Program Address
PC	Program Counter
PD	Program Data
PD	Power Domain
PHY	Physical Layer
PIN INT	PIN Interrupt
PLL	Phase Locked Loop
PPG	Programmable Pulse Generator
PPU	Peripheral Protection Unit
PSS	Power Saving State
PWM	Pulse Width Modulation
RAM	Random Access Memory
RICFG	Resource Input Configuration
RH	Remote Handler
RLT	Reload Timer
ROM	Read-Only Memory
RTC	Real Time Clock
RUN	RUN State
RX	Receive/Receiver
SA	Sector Address
SCT	Source Clock Timer
SDR	Security Description Record
SEC	Single Error Correction
SEC-DED ECC	Single Error Correcting, Double Error Detecting Error Correction Code
SGA	Sound Generator Amplitude Output
SGO	Sound Generator Tone Output
SHE	Secure Hardware Extension
SHECO	SHE COre
SMC	Stepper Motor Controller
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SRAM_IF	Static Random Access Memory Interface
SSCG	Spread Spectrum Clock Generator
TAP	Test Access Port
TB	Transaction Buffer

Table 1-3. Terms and abbreviations

Term	Meaning
TCFLASH	Tightly Coupled Flash
TCM	Tightly Coupled Memory
TF	Transaction Frame
TPU	Timing Protection Unit
TTCAN	Time Triggered CAN
TX	Transmit/Transmitter
UDC	Up/Down Counter
VIC	Vectored Interrupt Controller
VLAN	Virtual Local Area Network
Word	32 bits of data

1.5 Feature list

This section describes the list of features of FCR4 Cluster series.

Table 1-4. Features

Feature	Description
Technology	■ 90 nm CMOS with embedded flash
Processor Subsystem	■ Cortex R4 CPU core ■ 32-bit Arm architecture, dual-issue superscalar eight stage pipeline ■ Armv7 and Thumb-2 instruction set compliant ■ Memory Protection Unit (MPU) with 12 regions ■ Two Tightly Coupled Memory (TCM) ports. 64-bit AXI slave port for access to TCMs ■ 64-bit AXI master port ■ Vectored Interrupt Controller (VIC) port for faster interrupt processing ■ Single error correction, double error detection (SECDED) Error Correction Coding (ECC) for memory error detection and correction ■ Instruction cache: 16 KB 4-way set-associative ■ Data cache: 16 KB 4-way set-associative ■ Up to 8 break-points and 8 watchpoints
Debug and Trace	■ Arm Coresight technology ■ Standard 5-pin JTAG interface ■ 4-bit, 8-bit, 16-bit, and 32-bit trace data width supported depending on package ■ Secure entry supported for debugger
Graphics Sub-system	■ Refer to separate manual
Clocks	■ External main clock of 4 MHz (up to 8 MHz under evaluation) ■ External sub clock (typical 32.768 kHz) ■ Embedded RC oscillator (typical 8/12 MHz, configurable) ■ Embedded Slow RC oscillator (typical 100 kHz) ■ On-chip Phase Locked Loop (PLL) clock multiplier for main clock, Spread Spectrum Clock Generation (SSCG), and graphics ■ Stabilization timers for all source clocks
Clock Supervisor	■ Clock supervision for all source clocks and PLL outputs ■ Reset generation for out-of-bound clock frequencies on input source clocks, or PLL output clocks
Resets	■ Power on Reset (PoR) ■ External Reset ■ Software triggered hard reset ■ Clock supervision resets ■ Watchdog ■ Low Voltage Detection reset ■ Software reset

Table 1-4. Features

Feature	Description
Watchdog Timer	■ 32-bit counter ■ Supports selection of four clock sources (Main clock, Sub clock, RC clock, or Slow RC clock) ■ Support for window watchdog functionality ■ Reset or NMI generation support on watchdog errors ■ Support for preemptive warning interrupt before watchdog reset or NMI generation ■ Additional safety provision through three times redundancy and error correction logic for important configuration bits ■ Option to halt watchdog counter in case of core reaching break-point
DMA	■ 32-bit AHB interface ■ Block, burst, and demand transfer modes ■ Fixed and incremental addressing for source as well as destination ■ 8 channels to handle independent data flows ■ Fixed priority, dynamic priority, and round robin arbitration
Interrupts	■ Interrupt Request (IRQ) and Fast Interrupt Request (FIQ) capability ■ NMI sources generate FIQ ■ Supports request for low power mode entry ■ Programmable 32-level priority controller for normal IRQ sources. Also, supports programmable priority level masking ■ Programmable 16-level priority controller for NMI interrupt sources ■ Software interrupt generation ■ Privileged mode support for restricted access
External Interrupts	■ Up to 32 pins can be used as external interrupts ■ Optional 25 ns (typical) noise filters on all lines ■ DMA support ■ NMI support ■ Five polarity support (‘H’, ‘L’, rising edge, falling edge, and/any edge) ■ Event capture support for all 32 external interrupt pins ■ Software enabled monitoring of external events, with sampling frequency of 500 Hz to 16 MHz
Timing Protection	■ Up to eight identical 24-bit timers for execution time protection, locking time protection, inter-arrival time protection, or deadline protection ■ Normal and overflow mode support ■ Global linear prescaler (1 to 64) to scale down clock frequency ■ Additional, individual timer prescaler to support four different software programmable frequencies (1, 1/2, 1/4, and 1/16) ■ Start, stop, and continue options per timer controllable by software
Memory Protection	■ Memory protection unit for all bus masters ■ AXI interface support ■ 8 programmable memory regions, and one background region which covers entire 4 GB address space ■ Unauthorized access generates NMI

Table 1-4. Features

Feature	Description
Peripheral Protection	■ Protection to all peripherals and General Purpose IOs (GPIO) ■ Individual protection setting per peripheral and per GPIO channel ■ DMA access support for faster register configuration
CAN	■ Supports CAN protocol version 2.0 part A and B ■ Bit rates up to 1 Mbps ■ 64 message objects ■ Each message object has its own identifier mask ■ Programmable FIFO mode (concatenation of message objects) ■ Maskable interrupt ■ Disabled automatic retransmission mode for time triggered CAN applications ■ Programmable loop-back mode for self-test operation
USART/LIN	■ Programmable LIN or USART function ■ Full-duplex support ■ Clock synchronous (start-stop synchronization and start-stop-bit option), and Clock asynchronous (using start-, stop-bits) transfer modes ■ Dedicated baud rate generator. Mechanism for automatic baud rate adjust available in LIN mode ■ Support for data length of 7 bits (not in synchronous or LIN mode) and 8 bits ■ Support for signal modes Non-Return to Zero (NRZ) and Non-Return to Zero Inverted (NRZI) ■ Interrupt capability for transmission, reception, and errors ■ DMA support
I2C	■ Master/slave transmitting and receiving functions ■ 7-bit addressing as master and slave ■ 10-bit addressing as master and slave ■ Acknowledge disable option upon slave address reception (master-only operation) ■ Address masking to give interface several slave addresses ■ Up to 400 kbit transfer rate ■ Optional noise filters for SDA and SCL ■ Interrupt capability on transmission and bus error events
Secure Hardware Extension (SHE)	■ Provides cryptographic functions and secure key storage ■ Provides AES-128 encryption and decryption operations ■ Supports the generation of the cipher-based message authentication code ■ Provides a random number generation function ■ It has a unique read-only Identification item (UID) ■ It has a data interface for direct access to microcontroller memory locations
Stepper Motor Control	■ PWM duty cycle programmable from 0% to 100% ■ Programmable setting to select ‘L’, ‘H’, ‘PWM’, and ‘HighZ’ output ■ High current output pins

Table 1-4. Features

Feature	Description
A/D Converter	■ 32 channels ■ 26 channels are directly connected to pins ■ 6 channels are corresponding to Stepping Motor Controllers: Each channel is multi-plexed to one of the four pins of a Stepping Motor Controller ■ Conversion time: 1 μs per channel ■ RC type Successive Approximation (SAR) with sample and hold circuit ■ 10-bit or 8-bit resolution ■ Program selection analog input from 32 channels ■ Single conversion, continuous conversion, and scan conversion options ■ Interrupt capability ■ DMA support ■ Four range comparator channels for comparing conversion output with thresholds
I2S	■ Programmable master/slave operations ■ Supports transmission only, reception only and simultaneous transmission/ reception operations ■ Support for 1 sub frame and 2 sub frame constructions ■ Up to 32 channels supported in each sub frame ■ Support for individual configuration of channel number, channel length, word length in each sub frame ■ Word length support from 7 bits to 32 bits ■ Programmable frequency, polarity, and phase of frame synchronous signal ■ Programmable sampling point of received data (center or at the end of received data) ■ Support for frequency division from 1 to 126 in multiples of 2 ■ DMA support ■ Interrupt capability
Sound Generator	■ Produces sound/melody with varying frequency and amplitude ■ Square wave sound output with frequency of 100 Hz – 6 kHz (resolution 20 Hz) ■ Programmable Pulse Width Modulated (PWM) cycle width of 255 or 511 clocks. PWM duty cycle programmable from 0% to 100% ■ Two 2-bit prescaler with programmable clock division of 1, 1/2, 1/3, and1/4 ■ Automatic linear or exponential amplitude increment or decrement ■ Start, stop, resume functionality ■ DMA support ■ Automatic sound output stop when amplitude becomes ‘0’
Up Down Counter	■ One 32-bit channel or two 16-bit channels ■ Three count modes (timer mode, up/down count mode, and phase difference count mode) supported ■ Multiply by 2 or multiply by 4 in phase difference count mode ■ Count source can be internal clock or external trigger ■ Counting range: any value between 0 and 2^{32}-1 can be set ■ Four interrupt options (Compare-match interrupt, Underflow interrupt, Overflow interrupt, and Count direction change interrupt

Table 1-4. Features

Feature	Description
Reload Timers	■ 32-bit reload counter ■ External and Internal clock/event source ■ Trigger signal programmable as rising/falling edge or both ■ Gated count function ■ One-shot or reload counter mode ■ Counter state can be made visible at external pin ■ Prescaler with six different settings for the internal clock and two settings for the external clock ■ Several Reload Timers can be cascaded to form a longer Reload Timer ■ DMA support
Free Running Timers	■ Signals an interrupt on overflow, match with Compare registers, zero- detection, or match with Compare Clear Register ■ Option to mask zero detection, compare clear match interrupt, or both to allow for interrupt generation only after multiple events ■ Programmable timer period up to 1 sec at 64MHz input clock ■ Support for 11 counter clocks. Prescaler with 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, and 1/1024 of peripheral clock frequency ■ DMA support
Input Capture Units	■ Consists of 2 independent input channels ■ 16-bit wide capture registers per channel ■ Signals an interrupt upon external event ■ Rising edge, falling edge or rising, and falling edge sensitive ■ DMA support
Output Compare Units	■ Consists of 2 independent channels ■ 16-bit wide ■ Signals an interrupt when a match with 16-bit I/O timer occurs ■ A pair of compare registers can be used to generate an output signal ■ Interrupt capability
Programmable Pulse Generator	■ 16-bit down counter, cycle and duty setting registers ■ Interrupt at trigger, counter borrow and/or duty match ■ PWM operation and one-shot operation ■ Internal prescaler allows 1, 1/4, 1/16, 1/64 of peripheral clock as counter clock and Reload timer underflow as clock input ■ Can be triggered by software or reload timer
Real Time Clock	■ Can be clocked from Main clock, Sub clock, or RC clock ■ Automatic calibration support even when device is in low power state ■ Interrupt capability on half-second, 1 second, 1 minute, 1 hour, and 1 day duration ■ Additional capability for interrupt generation on calibration failure detection and calibration done event ■ Auto calibration of Sub clock or RC clock with respect to Main clock ■ Separate clock selector for calibration ■ Configurable calibration duration ■ Auto/manual trigger for calibration

Table 1-4. Features

Feature	Description
Internal memories- TCM-RAM	■ 128 KB ■ 64-bit interface ■ Single error correction, double error detection (SECDED) ECC support
Internal memories- System RAM	■ 64-bit AXI interface ■ 48 KB ■ Single error correction, double error detection (SECDED) ECC support ■ Parallel read/write capability for two different banks
Internal memories- Retention RAM	■ 16 KB ■ 4 banks ■ 32-bit AHB ■ Low leakage RAMs for low power consumption
Tightly Coupled Flash memory	■ up to 2 MB + 128 KB ■ Parallel Programming support ■ Mapped to TCM address space as well as Cacheable address space through AXI interface ■ Single error correction, double error detection (SECDED) ECC support ■ TCM address space supports only read access ■ Cacheable AXI address space supports write and read access ■ Detection of hang-up 1 state up to 8 large sectors of 128 KB or 256 KB each up to 8 small sectors of 16 KB each ■ Sector-wise access protection for write and read accesses
EEPROM Emulation Flash memory	■ 64 KB ■ Parallel Programming support ■ Single error correction, double error detection (SECDED) ECC support ■ Support for sector erase ■ EEPROM emulation mode support ■ Support for mirroring of memory in 3 diverse memory-mapped regions ■ 8 sectors of 8 KB each ■ Sector-wise access protection for write and read accesses
Quad SPI	■ Supports legacy as well as the dual-bit and quad-bit modes of SPI operation ■ Supports up to four slave devices in master mode ■ Programmable transfer rate, active-level of slave-select signal, polarity, and phase of the serial clock per slave select ■ Support for memory mapped operation of external serial flash and serial SRAM devices in command sequencer mode ■ Additional direct mode support for standard SPI operation through FIFO interface
Error Collection	■ Error collection on all peripherals ■ Optional Non-Maskable Interrupt (NMI) generation capability
Low Voltage Detect	■ Low voltage detection for 5 V, 3.3 V, and 1.2 V ■ Programmable thresholds ■ Reset generation capability on low voltage events

Table 1-4. Features

Feature	Description
I/O ports	■ All functional pins can be used as GPIO ■ Programmable analog or digital functionality selection ■ Programmable input levels (automotive, CMOS, and TTL) ■ Programmable pull-up/pull-down and output drive
Automotive Remote Handler	■ The integrated ARH controller provides 2 links ■ Both links can be connected over AIC to an external AShell ■ One link can connect over APIX® PHY ■ APIX® low bandwidth mode 1 and 2 are supported
External Bus Interface	■ Provides 16-bit data and 24-bit address width
MediaLB interface	■ Implements all basic functionality of a MediaLB Device, including ■ Transmission of Commands and Data when functioning as the transmitting Device associated with a ChannelAdress ■ Reception of Data and transmission of RxStatus responses when functioning as the receiving Device associated with a ChannelAddress ■ MediaLB lock detection ■ System Channel command handling ■ 3-pin MediaLB Mode is supported ■ Capable of running at speeds up to 1024Fs ■ Support for up to 31 logical channels with up to 124 bytes of data per frame

1.6 Memory map

This section describes the memory map of FCR4 Cluster device.

Table 1-5. FCR4 Cluster memory map

FFFF2000	Reserved
FFFF0000	BOOTROM
FFFEFC00	EXCFG
FFFEF800	NMIVAS
B0D01000	Reserved
B0D00000	SYSTEM_RAM_CONFIG
B0C00000	PERI5_AHB
B0B00000	PERI4_SLAVE
B0A00000	PERI3_ERBUS
B0900000	Reserved
B0800000	PERI1_RBUS
B0700000	PERI0_RBUS
B0600000	MCU_CONFIG
B0500000	DEBUG_BUS
B0400000	MEMORY_CONFIG
B0200000	Reserved
B0180000	EBICFG
B0100000	GFXCFG
B0080000	Reserved
B0000000	HSSPI0CFG
90000000	Reserved
80000000	HSSPI0_MEMORY
60000000	Reserved
40000000	GFXMEM
30000000	Reserved
20000000	EBI_MEMORY1
10000000	EBI_MEMORY0
06000000	Reserved
05FE0000	AXI_SLAVE_CORE0_TCM_FLASH_SMALL_SECTORS
05A00000	Reserved
05800000	AXI_SLAVE_CORE0_TCM_FLASH_LARGE_SECTORS
05020000	Reserved
05000000	AXI_SLAVE_CORE0_TCM_RAM
04800000	AXI_SLAVE_CORE0_DCACHE
04000000	AXI_SLAVE_CORE0_ICACHE
01A10000	Reserved
01A00000	SYSTEM_RAM
01800000	Reserved
017E0000	AXI_FLASH_MEMORY_SMALL_SECTORS
01200000	Reserved
01000000	AXI_FLASH_MEMORY_LARGE_SECTORS
00FE0000	TCM_FLASH_SMALL_SECTORS
00A00000	Reserved
00800000	TCM_FLASH_LARGE_SECTORS
00020000	Reserved
00000000	TCM_RAM

Table 1-6. PERI0_RBUS memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B07FFC00	BSU0	Yes	Yes	Yes	No	No
B07F8000	RICFG0	Yes	Yes	Yes	No	No
B07F0400	Reserved					
B07F0000	BECU0	Yes	Yes	Yes	No	No
B07EC000	Reserved					
B07E8000	PPC	Yes	Yes	Yes	No	No
B074C400	Reserved					
B074C000	PPGGLC0	Yes	Yes	Yes	No	No
B0749000	Reserved					
B0748C00	PPGGRP3	Yes	Yes	Yes	No	No
B0748800	PPGGRP2	Yes	Yes	Yes	No	No
B0748400	PPGGRP1	Yes	Yes	Yes	No	No
B0748000	PPGGRP0	Yes	Yes	Yes	No	No
B073C000	Reserved					
B073BC00	PPG15	Yes	Yes	Yes	No	No
B073B800	PPG14	Yes	Yes	Yes	No	No
B073B400	PPG13	Yes	Yes	Yes	No	No
B073B000	PPG12	Yes	Yes	Yes	No	No
B073AC00	PPG11	Yes	Yes	Yes	No	No
B073A800	PPG10	Yes	Yes	Yes	No	No
B073A400	PPG9	Yes	Yes	Yes	No	No
B073A000	PPG8	Yes	Yes	Yes	No	No
B0739C00	PPG7	Yes	Yes	Yes	No	No
B0739800	PPG6	Yes	Yes	Yes	No	No
B0739400	PPG5	Yes	Yes	Yes	No	No
B0739000	PPG4	Yes	Yes	Yes	No	No
B0738C00	PPG3	Yes	Yes	Yes	No	No
B0738800	PPG2	Yes	Yes	Yes	No	No
B0738400	PPG1	Yes	Yes	Yes	No	No
B0738000	PPG0	Yes	Yes	Yes	No	No
B0731C00	Reserved					
B0731800	SMCTG0	Yes	Yes	Yes	No	No
B0731400	SMC5	Yes	Yes	Yes	No	No
B0731000	SMC4	Yes	Yes	Yes	No	No
B0730C00	SMC3	Yes	Yes	Yes	No	No
B0730800	SMC2	Yes	Yes	Yes	No	No
B0730400	SMC1	Yes	Yes	Yes	No	No
B0730000	SMC0	Yes	Yes	Yes	No	No

Table 1-6. PERI0_RBUS memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B0728400	Reserved					
B0728000	USART0	Yes	Yes	Yes	No	No
B0720400	Reserved	Yes	Yes	Yes	No	No
B0720000	I2C0	Yes	Yes	Yes	No	No
B0718800	Reserved	Yes	Yes	Yes	No	No
B0718400	OCU1	Yes	Yes	Yes	No	No
B0718000	OCU0	Yes	Yes	Yes	No	No
B0711000	Reserved	Yes	Yes	Yes	No	No
B0710C00	ICU3	Yes	Yes	Yes	No	No
B0710800	ICU2	Yes	Yes	Yes	No	No
B0709000	Reserved	Yes	Yes	Yes	No	No
B0708C00	FRT3	Yes	Yes	Yes	No	No
B0708800	FRT2	Yes	Yes	Yes	No	No
B0708400	FRT1	Yes	Yes	Yes	No	No
B0708000	FRT0	Yes	Yes	Yes	No	No
B0700C00	Reserved					
B0700800	ADC1	Yes	Yes	Yes	No	No
B0700400	Reserved					
B0700000	ADC0	Yes	Yes	Yes	No	No

Table 1-7. PERI1_RBUS memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B08FFC00	BSU1	Yes	Yes	Yes	No	No
B08F8000	RICFG1	Yes	Yes	Yes	No	No
B08F0400	Reserved					
B08F0000	BECU1	Yes	Yes	Yes	No	No
B085C400	Reserved					
B085C000	PPGGLC1	Yes	Yes	Yes	No	No
B0858800	Reserved					
B0858400	PPGGRP17	Yes	Yes	Yes	No	No
B0858000	PPGGRP16	Yes	Yes	Yes	No	No
B084A000	Reserved					
B0849C00	PPG71	Yes	Yes	Yes	No	No
B0849800	PPG70	Yes	Yes	Yes	No	No
B0849400	PPG69	Yes	Yes	Yes	No	No
B0849000	PPG68	Yes	Yes	Yes	No	No
B0848C00	PPG67	Yes	Yes	Yes	No	No
B0848800	PPG66	Yes	Yes	Yes	No	No

Table 1-7. PERI1_RBUS memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B0848400	PPG65	Yes	Yes	Yes	No	No
B0848000	PPG64	Yes	Yes	Yes	No	No
B0838400	Reserved					
B0838000	USART6	Yes	Yes	Yes	No	No
B0828800	Reserved					
B0828400	OCU17	Yes	Yes	Yes	No	No
B0828000	OCU16	Yes	Yes	Yes	No	No
B0821000	Reserved					
B0820C00	ICU19	Yes	Yes	Yes	No	No
B0820800	ICU18	Yes	Yes	Yes	No	No
B0819000	Reserved					
B0818C00	FRT19	Yes	Yes	Yes	No	No
B0818800	FRT18	Yes	Yes	Yes	No	No
B0818400	FRT17	Yes	Yes	Yes	No	No
B0818000	FRT16	Yes	Yes	Yes	No	No
B0808C00	Reserved					
B0808800	CAN2	Yes	Yes	Yes	No	No
B0808400	CAN1	Yes	Yes	Yes	No	No
B0808000	CAN0	Yes	Yes	Yes	No	No
B0800400	Reserved					
B0800000	SG0	Yes	Yes	Yes	No	No

Table 1-8. PERI3_eRBUS memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B0AFFC00	BSU3	Yes	Yes	Yes	Yes	No
B0AF8000	RICFG3	Yes	Yes	Yes	Yes	No
B0AF0400	Reserved					
B0AF0000	BECU3	Yes	Yes	Yes	Yes	No
B0A20400	Reserved					
B0A20000	UDC0	Yes	Yes	Yes	Yes	No
B0A12800	Reserved					
B0A12400	RLT9	Yes	Yes	Yes	Yes	No
B0A12000	RLT8	Yes	Yes	Yes	Yes	No
B0A11C00	RLT7	Yes	Yes	Yes	Yes	No
B0A11800	RLT6	Yes	Yes	Yes	Yes	No
B0A11400	RLT5	Yes	Yes	Yes	Yes	No
B0A11000	RLT4	Yes	Yes	Yes	Yes	No
B0A10C00	RLT3	Yes	Yes	Yes	Yes	No

Table 1-8. PERI3_eRBUS memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B0A10800	RLT2	Yes	Yes	Yes	Yes	No
B0A10400	RLT1	Yes	Yes	Yes	Yes	No
B0A10000	RLT0	Yes	Yes	Yes	Yes	No
B0A09000	Reserved					
B0A08000	GPIO	Yes	Yes	Yes	Yes	No
B0A00400	Reserved					
B0A00000	PPU0	Yes	Yes	Yes	Yes	No

Table 1-9. PERI4_SLAVE AHB bus memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B0BFFC00	BSU4	Yes	Yes	Yes	No	No
B0BF8000	RICFG4	Yes	Yes	Yes	No	No
B0B40400	Reserved					
B0B40000	ARH0	Yes	Yes	Yes	No	No
B0B38C00	Reserved					
B0B38800	SPI2	Yes	Yes	Yes	No	No
B0B38400	SPI1	Yes	Yes	Yes	No	No
B0B38000	SPI0	Yes	Yes	Yes	No	No
B0B30400	Reserved					
B0B30000	CRC0	Yes	Yes	Yes	No	No
B0B20800	Reserved					
B0B20400	I2S1	Yes	Yes	Yes	No	No
B0B20000	I2S0	Yes	Yes	Yes	No	No
B0B10800	Reserved					
B0B10400	MPUHMLB0	Yes	Yes	Yes	No	No
B0B10000	Reserved					
B0B0C000	ETHRAM0	Yes	Yes	Yes	No	No
B0B08800	Reserved					
B0B08400	ERCFG0	Yes	Yes	Yes	No	No
B0B08000	ETH0	Yes	Yes	Yes	No	No
B0B00400	Reserved					
B0B00000	MPUXGFX	Yes	Yes	Yes	No	No

Table 1-10. PERI5_AHB bus memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B0700000	PERI0_RBUS	Yes	Yes	Yes	No	No
B0800000	PERI1_RBUS	Yes	Yes	Yes	No	No
B0CFFC00	BSU5	Yes	Yes	Yes	No	No
B0C09000	Reserved					
B0C08000	MPUXDMA0	Yes	No	Yes	No	No
B0C04000	Reserved					
B0C00000	DMA0	Yes	No	Yes	No	No

Table 1-11. Memory and config (MEMORY_CONFIG) AHB bus memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B04C0000	EEFLASH_NOECC_MIR	Yes	Yes	Yes	Yes	No
B0480000	EEFLASH_TABLE_MIR	Yes	Yes	Yes	Yes	No
B0440000	EEFLASH_ECC_MIR	Yes	Yes	Yes	Yes	No
B0418400	Reserved					
B0418000	BSU6	Yes	Yes	Yes	Yes	No
B0412400	Reserved					
B0412000	EEFCFG	Yes	Yes	Yes	Yes	No
B0411400	Reserved					
B0411000	TCFCFG	Yes	Yes	Yes	Yes	No
B0410400	Reserved					
B0410000	TRCFG	Yes	Yes	Yes	Yes	No
B0408400	Reserved					
B0408000	TPU0	Yes	Yes	Yes	Yes	No
B0401000	Reserved					
B0400000	IRQ0	Yes	Yes	Yes	Yes	No

Table 1-12. MCU_CONFIG AHB bus memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugging	Subsystem	SHE
B06FFC00	BSU7	Yes	Yes	Yes	No	No
B06F8000	RICFG7	Yes	Yes	Yes	No	No
B063C000	Reserved					
B063B000	RETRAMBANK3	Yes	Yes	Yes	No	Yes
B063A000	RETRAMBANK2	Yes	Yes	Yes	No	Yes
B0639000	RETRAMBANK1	Yes	Yes	Yes	No	Yes

Table 1-12. MCU_CONFIG AHB bus memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugging	Subsystem	SHE
B0638000	RETRAMBANK0	Yes	Yes	Yes	No	Yes
B0628400	Reserved					
B0628000	EICU0	Yes	Yes	Yes	No	No
B0620400	Reserved					
B0620000	EIC0	Yes	Yes	Yes	No	No
B0618400	Reserved					
B0618000	RTC	Yes	Yes	Yes	No	No
B0610400	Reserved					
B0610000	RRCFG	Yes	Yes	Yes	No	No
B0608400	Reserved					
B0608000	WDG	Yes	No	Yes	No	No
B0601000	Reserved					
B0600000	SYSC	Yes	No	Yes	No	No

Table 1-13. EBICFG memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B01FFC00	BSU10	Yes	Yes	Yes	Yes	No
B0180400	Reserved					
B0180000	EBI	Yes	Yes	Yes	Yes	Yes

Table 1-14. HSSPI0CFG memory map showing which masters can access which modules

Start address	Module	CPU	DMA	Debugger	Subsystem	SHE
B007FC00	BSU8	Yes	Yes	Yes	Yes	No
B0078000	RICFG8	Yes	Yes	Yes	Yes	No
B0000400	Reserved					
B0000000	HSSPI0	Yes	Yes	Yes	Yes	Yes

1.7 Register access

This section describes the registers that have a combined accessor and those that can be accessed using individual bit fields.

1.7.1 Combined registers

The following table is the list of registers that also has a combined accessor in the header file.

Table 1-15. Combined registers

BECUn_ADDRL	+ BECUn_ADDRH	BECUn_ADDR (32-bit)
BECUn_DATALL	+ BECUn_DATALH + BECUn_DATAHL + BECUn_DATAHH	BECUn_DATA (64-bit)
BECUn_MIDL	+ BECUn_MIDH	BECUn_MID (32-bit)

USARTn_BGRLL	+ USARTn_BGRLM + USARTn_BGRLH	USARTn_BGRL (32-bit)
USARTn_BGRL	+ USARTn_BGRM + USARTn_BGRH	USARTn_BGR (32-bit)

ADCn_ER32	+ ADCn_ER10	ADCn_ER (32-bit)
ADCn_RCOIRS32	+ ADCn_RCOIRS10	ADCn_RCOIRS (32-bit)
ADCn_RCOINT32	+ ADCn_RCOINT10	ADCn_RCOINT (32-bit)
ADCn_RCOOF32	+ ADCn_RCOOF10	ADCn_RCOOF (32-bit)
ADCn_RCOINTC32	+ ADCn_RCOINTC10	ADCn_RCOINTC (32-bit)
ADCn_PCZF32	+ ADCn_PCZF10	ADCn_PCZF (32-bit)
ADCn_PCZFC32	+ADCn_PCZFC10	ADCn_PCZFC (32-bit)
ADCn_PCIE32	+ ADCn_PCIE10	ADCn_PCIE (32-bit)
ADCn_PCIES32	+ ADCn_PCIES10	ADCn_PCIES (32-bit)
ADCn_PCIEC32	+ ADCn_PCIEC10	ADCn_PCIEC (32-bit)

BSU0_BTSTL	+ BSU0_BTSTH	BSU0_BTST (32-bit)
BSU1_BTSTL	+ BSU1_BTSTH	BSU1_BTST (32-bit)

CANn_IFxMSK1	+ CANn_IFxMSK2	CANn_IFxMSK (32-bit)
CANn_IFxARB1	+ CANn_IFxARB2	CANn_IFxARB (32-bit)

(all BSU0_PEN??L + BSU0_PEN??H -> BSU0_PEN?? registers)

1.7.2 Single bit access registers

The following is the list of available single bit registers.

Table 1-16. Single bit access registers

IRQn_NMIS:NMIS[31:0]	IRQn_NMIS:NMIS31	... IRQn_NMIS:NMIS0
IRQn_NMIR:NMIR[31:0]	IRQn_NMIR:NMIR31	... IRQn_NMIR:NMIR0
IRQn_NMISIS:NMISIS[31:0]	IRQn_NMISIS:NMISIS31	... IRQn_NMISIS:NMISIS0
IRQn_IRQS0:IRQS[31:0]	IRQn_IRQS0:IRQS31	... IRQn_IRQS0:IRQS0
IRQn_IRQR0~15:IRQR[31:0]	IRQn_IRQR0~15:IRQR31	... IRQn_IRQR0~15:IRQR0
IRQn_IRQSIS0~15:IRQSIS[31:0	IRQn_IRQSIS0~15:IRQSIS31	... IRQn_IRQSIS0~15:IRQSIS0
IRQn_IRQCES0~15:IRQCES[31:0]	IRQn_IRQCES0~15:IRQCES31	.. IRQn_IRQCES0~15:IRQCES0
IRQn_IRQCEC0~15:IRQCEC[31:0]	IRQn_IRQCEC0~15:IRQCEC31	... IRQn_IRQCEC0~15:IRQCEC0
IRQn_IRQCE0~15:IRQCE[31:0]	IRQn_IRQCE0~15:IRQCE31	... IRQn_IRQCE0~15:IRQCE0
IRQn_NMIHS:NMIHS[31:0]	IRQn_NMIHS:NMIHS31	... IRQn_NMIHS:NMIHS0
IRQn_IRQHS0~15:IRQHS[31:0]	IRQn_IRQHS0~15:IRQHS31	... IRQn_IRQHS0~15:IRQHS0
IRQn_NMIRS:NMIRS[31:0]	IRQn_NMIRS:NMIRS31	... IRQn_NMIRS:NMIRS0
IRQn_NMIPS:NMIPS[31:0]	IRQn_NMIPS:NMIPS31	... IRQn_NMIPS:NMIPS0
IRQn_IRQRS0~15:IRQRS[31:0]	IRQn_IRQRS0~15:IRQRS31	... IRQn_IRQRS0~15:IRQRS0
IRQn_IRQPS0~15:IRQPS[31:0]	IRQn_IRQPS0~15:IRQPS31	... IRQn_IRQPS0~15:IRQPS0

EICUn_IRENR:IREN[31:0]	EICUn_IRENR:IREN31	... EICUn_IRENR:IREN0

TPUn_TIR:IR[7:0]	TPUn_TIR:IR7	... TPUn_TIR:IR0
TPUn_TST:TS[7:0]	TPUn_TST:TS7	... TPUn_TST:TS0
TPUn_TIE:IE[7:0]	TPUn_TIE:IE7	... TPUn_TIE:IE0

PPUn_PR0~15:PR[31:0]	PPUn_PR0~15:PR31	... PPUn_PR0~15:PR0
PPUn_PRS0~15:PRS[31:0]	PPUn_PRS0~15:PRS31	... PPUn_PRS0~15:PRS0
PPUn_PRC0~15:PRC[31:0]	PPUn_PRC0~15:PRC31	... PPUn_PRC0~15:PRC0
PPUn_PA0~15:PA[31:0]	PPUn_PA0~15:PA31	... PPUn_PA0~15:PA0
PPUn_PAS0~15:PAS[31:0]	PPUn_PAS0~15:PAS31	... PPUn_PAS0~15:PAC0
PPUn_PAC0~15:PAC[31:0]	PPUn_PAC0~15:PAC31	... PPUn_PAS0~15:PAC0
PPUn_GA0~15:GA[31:0]	PPUn_GA0~15:GA31	... PPUn_GA0~15:GA0
PPUn_GAS0~15:GAS[31:0]	PPUn_GAS0~15:GAS31	... PPUn_GAS0~15:GAC0
PPUn_GAC0~15:GAC[31:0]	PPUn_GAC0~15:GAC31	... PPUn_GAS0~15:GAC0

Table 1-16. Single bit access registers

EICn_ENIR:EN[31:0]	EICn_ENIR:EN31	... EICn_ENIR:EN0
EICn_ENISR:ENS[31:0]	EICn_ENISR:ENS31	... EICn_ENISR:ENS0
EICn_ENICR:ENC[31:0]	EICn_ENICR:ENC31	... EICn_ENICR:ENC0
EICn_EIRR:ER[31:0]	EICn_EIRR:ER31	... EICn_EIRR:ER0
EICn_EIRCR:ERC[31:0]	EICn_EIRCR:ERC31	... EICn_EIRCR:ERC0
EICn_NFER:NFE[31:0]	EICn_NFER:NFE31	... EICn_NFER:NFE0
EICn_NFESR:NFES[31:0]	EICn_NFESR:NFES31	... EICn_NFESR:NFES0
EICn_NFECR:NFEC[31:0]	EICn_NFECR:NFEC31	... EICn_NFECR:NFEC0
EICn_DRER:DRE[31:0]	EICn_DRER:DRE31	... EICn_DRER:DRE0
EICn_DRESR:DRES[31:0]	EICn_DRESR:DRES31	... EICn_DRESR:DRES0
EICn_DRECR:DREC[31:0]	EICn_DRECR:DREC31	... EICn_DRECR:DREC0
EICn_DRFR:DRF[31:0]	EICn_DRFR:DRF31	... EICn_DRFR:DRF0

PPGGRPp_GCTRL:EN[3:0]	PPGGRPp_GCTRL:EN3	... PPGGRPp_GCTRL:EN0
PPGGLCg_GCNR:CTG[1:0]	PPGGLCg_GCNR:CTG1	... PPGGLCg_GCNR:CTG0

I2Sn_MCR1REG:S0CH[31:0]	I2Sn_MCR1REG:S0CH31	... I2Sn_MCR1REG:S0CH0
I2Sn_MCR2REG:S1CH[31:0]	I2Sn_MCR2REG:S1CH31	... I2Sn_MCR2REG:S1CH0

GPIO_DDRn:DD[63:0]	GPIO_DDRn:DD63	... GPIO_DDRn:DD0
GPIO_DDSRn:DDS[63:0]	GPIO_DDSRn:DDS63	... GPIO_DDSRn:DDS0
GPIO_DDCRn:DDC[63:0]	GPIO_DDCRn:DDC63	... GPIO_DDCRn:DDC0
GPIO_PODRn:POD[63:0]	GPIO_PODRn:POD63	... GPIO_PODRn:POD0
GPIO_POSRn:POS[63:0]	GPIO_POSRn:POS63	... GPIO_POSRn:POS0
GPIO_POCRn:POC[63:0]	GPIO_POCRn:POC63	... GPIO_POCRn:POC0
GPIO_PIDRn:PID[63:0]	GPIO_PIDRn:PID63	... GPIO_PIDRn:PID0
GPIO_PPERn:PPE[63:0]	GPIO_PPERn:PPE63	... GPIO_PPERn:PPE0

CANn_TREQR1:TXRQST[16:1]	CANn_TREQR1:TXRQST1	... CANn_TREQR1:TXRQST16
CANn_TREQR2:TXRQST[32:17]	CANn_TREQR2:TXRQST17	... CANn_TREQR2:TXRQST32
CANn_TREQR3:TXRQST[48:33]	CANn_TREQR3:TXRQST33	... CANn_TREQR1:TXRQST48
CANn_TREQR4:TXRQST[64:49]	CANn_TREQR4:TXRQST49	... CANn_TREQR2:TXRQST64
CANn_NEWDT1:NEWDAT[16:1]	CANn_NEWDT1:NEWDAT1	... CANn_NEWDT1:NEWDAT16
CANn_NEWDT2:NEWDAT[32:17]	CANn_NEWDT2:NEWDAT17	... CANn_NEWDT2:NEWDAT32
CANn_NEWDT3:NEWDAT[48:33]	CANn_NEWDT1:NEWDAT33	... CANn_NEWDT1:NEWDAT48
CANn_NEWDT4:NEWDAT[64:49]	CANn_NEWDT2:NEWDAT49	... CANn_NEWDT2:NEWDAT64
CANn_INTPND1:INTPND[16:1]	CANn_INTPND1:INTPND1	... CANn_INTPND1:INTPND16
CANn_INTPND2:INTPND[32:17]	CANn_INTPND2:INTPND17	... CANn_INTPND2:INTPND32
CANn_INTPND3:INTPND[48:33]	CANn_INTPND1:INTPND33	... CANn_INTPND1:INTPND48
CANn_INTPND4:INTPND[64:49]	CANn_INTPND2:INTPND49	... CANn_INTPND2:INTPND64
CANn_MSGVAL1:MSGVAL[16:1]	CANn_MSGVAL1:MSGVAL1	... CANn_MSGVAL1:MSGVAL16
CANn_MSGVAL2:MSGVAL[32:17]	CANn_MSGVAL2:MSGVAL17	... CANn_MSGVAL2:MSGVAL32
CANn_MSGVAL3:MSGVAL[48:33]	CANn_MSGVAL1:MSGVAL33	... CANn_MSGVAL1:MSGVAL48
CANn_MSGVAL4:MSGVAL[64:49]	CANn_MSGVAL2:MSGVAL49	... CANn_MSGVAL2:MSGVAL64

Table 1-16. Single bit access registers

DMAn_DIRQ1:DIRQ[31:0]	DMAn_DIRQ1:DIRQ31	... DMAn_DIRQ1:DIRQ0
DMAn_DIRQ2:DIRQ[63:32]	DMAn_DIRQ2:DIRQ63	... DMAn_DIRQ2:DIR32
DMAn_EDIRQ1:EDIRQ[31:0]	DMAn_EDIRQ1:EDIRQ31	... DMAn_EDIRQ1:EDIRQ0
DMAn_EDIRQ2:EDIRQ[63:32]	DMAn_EDIRQ2:EDIRQ63	... DMAn_EDIRQ2:EDIRQ32

2. System Controller

This chapter explains the function and operation of the System Controller.

2.1 Outline of System Controller

This section describes the features and the block diagram of the System Controller.

Features of System Controller

The System Controller implements power, reset, and clock management in the device. This module also supports wakeup of the device from low power state. Low Voltage Detection on external power supply and clock supervision of the external clocks/PLL clocks are performed in this module. System Controller also has four Source Clock Timers, which are used for clock stabilization count of the source clock during oscillator initialization and as normal timer after clock stabilization. This module is in the always ON power domain and works on RC clock. System Controller can be configured through CPU. The following is a list of features:

- Embedded Slow RC and RC oscillator with control registers
- Main and Sub oscillator pad controllllllll
- Includes Main PLL for non-modulated clock generation
- Includes SSCG PLL for clock frequency modulation for EMI reduction
- Includes Graphics PLL for graphics support. This PLL clock is frequency modulated for EMI reduction
- Supports oscillator stabilization time for all clock source
- Clock generation and control, clock selection, and distribution to the device
- Support for device mode detection
- Reset management of the device
- Power management of the device
- Device state management
- Wakeup support from low power state
- Clock Supervision of Main oscillator, Sub oscillator, and PLL output clocks
- Low Voltage Detection
- Supports protection sequence for configuration registers programming
- Generates status interrupt for device state changes
- Generates error interrupt for device state profile related errors, clock errors, Low Voltage Detection

System Controller important key list

This section lists the keys used in the System Controller.

Table 2-1. Terms and abbreviations

Key value	Relevant register	Purpose
0x5CACCE55	SYSC_PROTKEYR:PROTKEY[31:0]	System Controller register unlock key value
0xAB	SYSC_TRGRUNCNTR:APPLYRUN[7:0]	RUN profile trigger value
0xBA	SYSC_PSSENR:PSSEN[7:0]	PSS enable register value

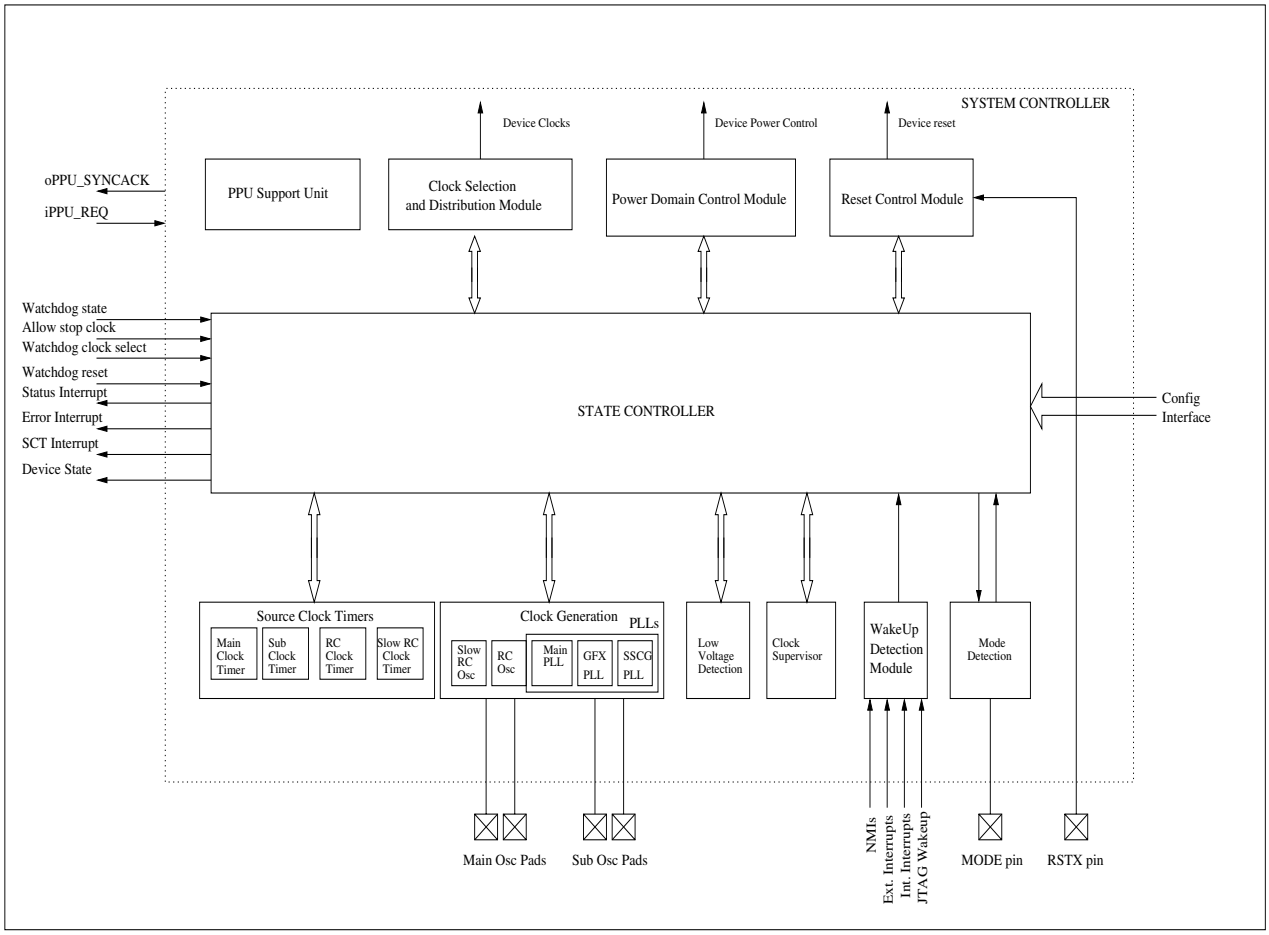
Table 2-1. Terms and abbreviations

Key value	Relevant register	Purpose
0x5A	SYSC_RSTCNTR:SWRST[7:0]	Software-triggered reset value
0xA5	SYSC_RSTCNTR:SWHRST[7:0]	Software-triggered hardware reset value
0xDA	SYSC_RSTCNTR:DBGR[7:0]	Debug reset value

Block diagram of System Controller

The Figure 2-1 shows the internal block diagram of the System Controller.

Figure 2-1. Block diagram of System Controller



Note: watchdog_clock_select signal indicates the clock selected for the watchdog operation. watchdog_state signal indicates watchdog enable/disable status in RUN and PSS state. allow_stop_clock is an indication from watchdog that device can enter into deep power down state in PSS where all the source clocks are cut. This signal enables the System Controller to cut the watchdog clock in PSS state of the device, even though Watchdog is enabled in PSS state.

iPPU_REQ and oPPU_SYNCACK signals are interfaced to PPU module. For details refer to PPU specification.

For more details on watchdog state handling in System Controller refer to 2.3.2.3 Device state switching, 2.3.2.7 Device state profiles, and 4. Watchdog Timer.

2.2 System Controller registers

This section lists all the System Controller registers and explains their function in detail.

The suffix ‘n’ in the register name indicates that the register is an instance ‘n’ of the module.

2.2.1 Registers of System Controller

The following registers are available for System Controller:

Protection register

- Protection Key Register (SYSC_PROTKEYR)

RUN profile register set

- RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR)
- RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER)
- RUN Profile Clock Select Register (SYSC_RUNCKSELR)
- RUN Profile Clock Enable Register (SYSC_RUNCKER)
- RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0)
- RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1)
- RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2)
- RUN Profile PLL Control Register (SYSC_RUNPLLCNTR)
- RUN Profile SSCG Control Register 0 (SYSC_RUNSSCGCNTR0)
- RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1)
- RUN Profile Graphics PLL Control Register 0 (SYSC_RUNGFXCNTR0)
- RUN Profile Graphics PLL Control Register 1 (SYSC_RUNGFXCNTR1)
- RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR)
- RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR)
- Trigger RUN Control Register (SYSC_TRGRUNCNTR)

PSS profile register set

- PSS Profile Power Domain Configuration Register (SYSC_PSSPDCFGR)
- PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER)
- PSS Profile Clock Select Register (SYSC_PSSCKSELR)
- PSS Profile Clock Enable Register (SYSC_PSSCKER)
- PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0)
- PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1)
- PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2)
- PSS Profile PLL Control Register (SYSC_PSSPLLCNTR)
- PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0)
- PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1)
- PSS Profile Graphics PLL Control Register 0 (SYSC_PSSGFXCNTR0)
- PSS Profile Graphics PLL Control Register 1 (SYSC_PSSGFXCNTR1)
- PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR)
- PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR)
- PSS Enable Register (SYSC_PSSENR)

APPLIED profile register set

- APPLIED Profile Power Domain Configuration Register (SYSC_APPPDCFGR)

- APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER)
- APPLIED Profile Clock Select Register (SYSC_APPCKSELR)
- APPLIED Profile Clock Enable Register (SYSC_APPCKER)
- APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0)
- APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1)
- APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2)
- APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR)
- APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0)
- APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1)
- APPLIED Profile Graphics PLL Control Register 0 (SYSC_APPGFXCNTR0)
- APPLIED Profile Graphics PLL Control Register 1 (SYSC_APPGFXCNTR1)
- APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR)
- APPLIED Profile Clock Supervisor Configuration Register (SYSC_APPCSVCFGR)

Profile status register set

- Power Domain Status Register (SYSC_PDSTSR)
- Clock Source Enable Status Register (SYSC_CKSRESTSR)
- Clock Select Status Register (SYSC_CKSELSTSR)
- Clock Enable Status Register (SYSC_CKESTSR)
- Clock Divider Register 0 (SYSC_CKDIVSTSR0)
- Clock Divider Register 1 (SYSC_CKDIVSTSR1)
- Clock Divider Register 2 (SYSC_CKDIVSTSR2)
- PLL Status Register (SYSC_PLLSTSR)
- SSCG PLL Status Register 0 (SYSC_SSCGSTSR0)
- SSCG PLL Status Register 1 (SYSC_SSCGSTSR1)
- Graphics PLL Status Register 0 (SYSC_GFXSTSR0)
- Graphics PLL Status Register (SYSC_GFXSTSR1)
- LVD Configuration Status Register (SYSC_LVDCFGSTSR)
- CSV Configuration Status Registe (SYSC_CSVCFGSTSR)

System registers

- System ID Register (SYSC_SYSIDR)
- System Status Register (SYSC_SYSSTSR)
- System Status Interrupt Enable Register (SYSC_SYSINTER)
- System Status Interrupt Clear Register (SYSC_SYSICLR)
- System Error Register (SYSC_SYSERRR)
- System Error Interrupt Clear Register (SYSC_SYSERRICLR)

Clock Supervisor Configuration registers

- Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR)
- Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR)
- Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR)
- Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR)
- Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR)
- Clock Supervisor Test Register (SYSC_CSVTESTR)

Reset control registers

- Reset Control Register (SYSC_RSTCNTR)
- User Reset Cause Register (SYSC_RSTCAUSEUR)
- BootROM Reset Cause Register (SYSC_RSTCAUSEBT)

Slow RC Source Clock Timer registers

- Slow RC SCT Trigger Register (SYSC_SRCSCTTRG)
- Slow RC SCT Control Register (SYSC_SRCSCTCNTR)
- Slow RC SCT Compare Prescaler Register (SYSC_SRCSCTCPR)
- Slow RC SCT Status Register (SYSC_SRCSCTSTATR)
- Slow RC SCT Interrupt Enable Register (SYSC_SRCSCTINTER)
- Slow RC SCT Interrupt Clear Register (SYSC_SRCSCTICLR)

RC Source Clock Timer registers

- RC SCT Trigger Register (SYSC_RCSCTTRG)
- RC SCT Control Register (SYSC_RCSCTCNTR)
- RC SCT Compare Prescaler Register (SYSC_RCSCTCPR)
- RC SCT Status Register (SYSC_RCSCTSTATR)
- RC SCT Interrupt Enable Register (SYSC_RCSCTINTER)
- RC SCT Interrupt Clear Register (SYSC_RCSCTICLR)

Main Source Clock Timer Registers

- Main SCT Trigger Register (SYSC_MAINSCTTRG)
- Main SCT Control Register (SYSC_MAINSCTCNTR)
- Main SCT Compare Prescaler Register (SYSC_MAINSCTCPR)
- Main SCT Status Register (SYSC_MAINSCTSTATR)
- Main SCT Interrupt Enable Register (SYSC_MAINSCTINTER)
- Main SCT Interrupt Clear Register (SYSC_MAINSCTICLR)

Sub Source Clock Timer Registers

- Sub SCT Trigger Register (SYSC_SUBSCTTRG)
- Sub SCT Control Register (SYSC_SUBSCTCNTR)
- Sub SCT Compare Prescaler Register (SYSC_SUBSCTCPR)
- Sub SCT Status Register (SYSC_SUBSCTSTATR)
- Sub SCT Interrupt Enable Register (SYSC_SUBSCTINTER)
- Sub SCT Interrupt Clear Register (SYSC_SUBSCTICLR)

Clock output function configuration registers

- Clock Output Function Configuration Register (SYSC_CKOTCFGR)

Special configuration registers

- Special Configuration Register (SYSC_SPCCFGR)
- RC Configuration Register (SYSC_RCCFGR)
- Test 0 Register (SYSC_TESTR0)
- Test 1 Register (SYSC_TESTR1)
- Test 2 Register (SYSC_TESTR2)

Debugger registers

- JTAG Detect Register (SYSC_JTAGDETECT)
- JTAG Configuration Register (SYSC_JTAGCNFG)
- JTAG Wakeup Register (SYSC_JTAGWAKEUP)

2.2.2 Memory layout of System Controller registers

Table 2-2. Memory layout of System Controller registers

Offset	+3	+2	+1	+0
0x00000000	SYSC_PROTKEYR 00000000 00000000 00000000 00000000			
0x00000004 - 0x0000007C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000080	SYSC_RUNCKSRER 00000000 00001110		SYSC_RUNPDCFGR 00000000 00001111	
0x00000084	SYSC_RUNCKSELR 00100011 00000000 00000000 00000000			
0x00000088	SYSC_RUNCKER 01110001 00111111 00001101 11110001			
0x0000008C	SYSC_RUNCKDIVR0 00000000 00000000 00000000 00000000			
0x00000090	SYSC_RUNCKDIVR1 00000000 00000000 00000000 00000000			
0x00000094	SYSC_RUNCKDIVR2 00000000 00000000 00000000 00000000			
0x00000098	SYSC_RUNPLLCNTR 00000000 00001101 00000001 00000000			
0x0000009C	SYSC_RUNSSCGCNTR0 00000001 00001101 00000001 00000000			
0x000000A0	SYSC_RUNSSCGCNTR1 00000000 00000000 00000000 00101001			
0x000000A4	SYSC_RUNGFXCNTR0 00000001 00001101 00000000 00000000			
0x000000A8	SYSC_RUNGFXCNTR1 00000000 00000000 00000000 00101001			
0x000000AC	SYSC_RUNLVDCFGR 00000000 00001011 00001110 00001111			
0x000000B0	SYSC_TRGRUNCNTR 00000000 00000000		SYSC_RUNCSVCFGR 00000000 00000000	
0x000000B4 - 0x000000FC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000100	SYSC_PSSCKSRER 00000000 00001110		SYSC_PSSPDCFGR 00000000 00001111	

Table 2-2. Memory layout of System Controller registers

Offset	+3	+2	+1	+0
0x00000104	SYSC_PSSCKSELR 00100011 00000000 00000000 00000000			
0x00000108	SYSC_PSSCKER 01110001 00111110 00001101 11110001			
0x0000010C	SYSC_PSSCKDIVR0 00000000 00000000 00000000 00000000			
0x00000110	SYSC_PSSCKDIVR1 00000000 00000000 00000000 00000000			
0x00000114	SYSC_PSSCKDIVR2 00000000 00000000 00000000 00000000			
0x00000118	SYSC_PSSPLLCNTR 00000000 00001101 00000001 00000000			
0x0000011C	SYSC_PSSSSCGCNTR0 00000001 00001101 00000001 00000000			
0x00000120	SYSC_PSSSSCGCNTR1 00000000 00000000 00000000 00101001			
0x00000124	SYSC_PSSGFXCNTR0 00000001 00001101 00000000 00000000			
0x00000128	SYSC_PSSGFXCNTR1 00000000 00000000 00000000 00101001			
0x0000012C	SYSC_PSSLVDCFGR 00000000 00001011 00001110 00001111			
0x00000130	SYSC_PSSENR 00000000 00000000		SYSC_PSSCSVCFGR 00000000 00000000	
0x00000134 - 0x0000017C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000180	SYSC_APPCKSRER 00000000 00001110		SYSC_APPPDCFGR 00000000 00001111	
0x00000184	SYSC_APPCKSELR 00100011 00000000 00000000 00000000			
0x00000188	SYSC_APPCKER 01110001 00111111 00001101 11110001			
0x0000018C	SYSC_APPCKDIVR0 00000000 00000000 00000000 00000000			
0x00000190	SYSC_APPCKDIVR1 00000000 00000000 00000000 00000000			
0x00000194	SYSC_APPCKDIVR2 00000000 00000000 00000000 00000000			
0x00000198	SYSC_APPPLLCNTR 00000000 00001101 00000001 00000000			
0x0000019C	SYSC_APPSSCGCNTR0 00000001 00001101 00000001 00000000			

Table 2-2. Memory layout of System Controller registers

Offset	+3	+2	+1	+0
0x000001A0	SYSC_APPSSCGCNTR1 00000000 00000000 00000000 00101001			
0x000001A4	SYSC_APPGFXCNTR0 00000001 00001101 00000000 00000000			
0x000001A8	SYSC_APPGFXCNTR1 00000000 00000000 00000000 00101001			
0x000001AC	SYSC_APPLVDCFGR 00000000 00001011 00001110 00001111			
0x000001B0	reserved 00000000 00000000		SYSC_APPCSVCFGR 00000000 00000000	
0x000001B4 - 0x000001FC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000200	SYSC_CKSRESTSR 00000000 00001110		SYSC_PDSTSR 00000000 00001111	
0x00000204	SYSC_CKSELSTSR 00100011 00000000 00000000 00000000			
0x00000208	SYSC_CKESTSR 01110001 00111111 00001101 11110001			
0x0000020C	SYSC_CKDIVSTSR0 00000000 00000000 00000000 00000000			
0x00000210	SYSC_CKDIVSTSR1 00000000 00000000 00000000 00000000			
0x00000214	SYSC_CKDIVSTSR2 00000000 00000000 00000000 00000000			
0x00000218	SYSC_PLLSTSR 00000000 00001101 00000001 00000000			
0x0000021C	SYSC_SSCGSTSR0 00000001 00001101 00000001 00000000			
0x00000220	SYSC_SSCGSTSR1 00000000 00000000 00000000 00101001			
0x00000224	SYSC_GFXSTSR0 00000001 00001101 00000000 00000000			
0x00000228	SYSC_GFXSTSR1 00000000 00000000 00000000 00101001			
0x0000022C	SYSC_LVDCFGSTSR 00000000 00101011 00101110 00101111			
0x00000230	reserved 00000000 00000000		SYSC_CSVCFGSTSR 00000000 00000000	
0x00000234 - 0x0000027C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000280	SYSC_SYSIDR XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			

Table 2-2. Memory layout of System Controller registers

Offset	+3	+2	+1	+0
0x00000284	SYSC_SYSSTSR 00000000 00000000 00000100 00000000			
0x00000288	SYSC_SYSINTER 00000000 00000000 00000000 00000000			
0x0000028C	SYSC_SYSICLR 00000000 00000000 00000000 00000000			
0x00000290	SYSC_SYSERRR 00000000 00000000 00000000 00000000			
0x00000294	SYSC_SYSERRICLR 00000000 00000000 00000000 00000000			
0x00000298 - 0x000002FC	reserved 00000000 00000000 00000000 00000000			
0x00000300	SYSC_CSVMOCFGR 00000000 00000000 00000000 00000000			
0x00000304	SYSC_CSVSOCFGR 00000000 00000000 00000000 00000000			
0x00000308	SYSC_CSVMPCFGR 00000000 00000000 00000000 00000000			
0x0000030C	SYSC_CSVSPCFGR 00000000 00000000 00000000 00000000			
0x00000310	SYSC_CSVGPCFGR 00000000 00000000 00000000 00000000			
0x00000314	SYSC_CSVTESTR 00000000 00000000 00000000 00000000			
0x00000318 - 0x0000037C	reserved 00000000 00000000 00000000 00000000			
0x00000380	SYSC_RSTCNTR 00000000 00000000 00000000 00000000			
0x00000384	SYSC_RSTCAUSEUR 00011110 00000000 00000000 00000001			
0x00000388	SYSC_RSTCAUSEBT X0011110 00000000 00000000 00000001			
0x0000038C - 0x000003FC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000400	SYSC_SRCSCTTRG 00000000 00000000 00000000 00000000			
0x00000404	SYSC_SRCSCTCNTR 00000000 00000000 00000000 00000000			
0x00000408	SYSC_SRCSCTCPR 00000000 00000110 00000000 00000001			
0x0000040C	SYSC_SRCSCTSTATR 00000000 00000000 00000000 00000000			

Table 2-2. Memory layout of System Controller registers

Offset	+3	+2	+1	+0
0x00000410	SYSC_SRCSCTINTER 00000000 00000000 00000000 00000000			
0x00000414	SYSC_SRCSCTICLR 00000000 00000000 00000000 00000000			
0x00000418 - 0x0000047C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000480	SYSC_RCSCTTRG 00000000 00000000 00000000 00000000			
0x00000484	SYSC_RCSCTCNTR 00000000 00000000 00000000 00000000			
0x00000488	SYSC_RCSCTCPR 00000000 00000110 00000000 00011110			
0x0000048C	SYSC_RCSCTSTATR 00000000 00000000 00000000 00000000			
0x00000490	SYSC_RCSCTINTER 00000000 00000000 00000000 00000000			
0x00000494	SYSC_RCSCTICLR 00000000 00000000 00000000 00000000			
0x00000498 - 0x000004FC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000500	SYSC_MAINSCTTRG 00000000 00000000 00000000 00000000			
0x00000504	SYSC_MAINSCTCNTR 00000000 00000000 00000000 00000000			
0x00000508	SYSC_MAINSCTCPR 00000000 00000110 00010000 00000000			
0x0000050C	SYSC_MAINSCTSTATR 00000000 00000000 00000000 00000000			
0x00000510	SYSC_MAINSCTINTER 00000000 00000000 00000000 00000000			
0x00000514	SYSC_MAINSCTICLR 00000000 00000000 00000000 00000000			
0x00000518 - 0x0000057C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000580	SYSC_SUBSCTTRG 00000000 00000000 00000000 00000000			
0x00000584	SYSC_SUBSCTCNTR 00000000 00000000 00000000 00000000			
0x00000588	SYSC_SUBSCTCPR 00000000 00000110 00000100 00000000			
0x0000058C	SYSC_SUBSCTSTATR 00000000 00000000 00000000 00000000			

Table 2-2. Memory layout of System Controller registers

Offset	+3	+2	+1	+0
0x00000590	SYSC_SUBSCTINTER 00000000 00000000 00000000 00000000			
0x00000594	SYSC_SUBSCTICLR 00000000 00000000 00000000 00000000			
0x00000598 - 0x000005FC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000600	SYSC_CKOTCFGR 00000000 00000000 00000000 00000111			
0x00000604 - 0x0000067C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000680	SYSC_SPCCFGR 00000000 000000X0 00000000 00000000			
0x00000684	SYSC_RCCFGR 00000000 00000000 00000001 11111111			
0x00000688	SYSC_TESTR0 00000000 00000000 00000000 00000000			
0x0000068C	SYSC_TESTR1 00000000 00000000 00000000 00000000			
0x00000690	SYSC_TESTR2 00000000 00000000 00000000 00000000			
0x00000694 - 0x000006FC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000700	SYSC_JTAGDETECT 00000000 00000000 00000000 00000000			
0x00000704	SYSC_JTAGCNFG 00000000 00000000 00000000 00000001			
0x00000708	SYSC_JTAGWAKEUP 00000000 00000000 00000000 00000001			
0x0000070C - 0x00000790	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			

2.2.3 Protection register

This register provides protection feature to System Controller registers from accidental writes. This register get initialized on hard resets.

2.2.3.1 Protection Key Register (SYSC_PROTKEYR)

Protection Key Register (SYSC_PROTKEYR) needs to be written with the correct key (0x5CACCE55) before each write access to other System Controller registers. However, there is an exception - access to the System Controller debug register does not require a protected sequence. Writing to this register with the correct key unlocks the write access to System Controller registers. Key is locked again after next write on AHB configuration bus. Wrong key write to this register, or writing to the System Controller register without unlocking, or writing two times the Protection Key Register causes error response on slave bus and hence data abort. This register should be written by only 32-bit writes.

Protection Key Register (SYSC_PROTKEYR)

Figure 2-2. Protection Key Register (SYSC_PROTKEYR)

SYSC_PROTKEYR																															
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**
PROTKEY[31]	PROTKEY[30]	PROTKEY[29]	PROTKEY[28]	PROTKEY[27]	PROTKEY[26]	PROTKEY[25]	PROTKEY[24]	PROTKEY[23]	PROTKEY[22]	PROTKEY[21]	PROTKEY[20]	PROTKEY[19]	PROTKEY[18]	PROTKEY[17]	PROTKEY[16]	PROTKEY[15]	PROTKEY[14]	PROTKEY[13]	PROTKEY[12]	PROTKEY[11]	PROTKEY[10]	PROTKEY[9]	PROTKEY[8]	PROTKEY[7]	PROTKEY[6]	PROTKEY[5]	PROTKEY[4]	PROTKEY[3]	PROTKEY[2]	PROTKEY[1]	PROTKEY[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-3. Protection Key Register (SYSC_PROTKEYR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PROTKEY	System Controller Protection Key Write key 0x5CACCE55 to unlock the System Controller registers for writing. Writing any other value to this register causes data abort. When System Controller registers are unlocked, read to this register returns 0xFFFFFFFF. In case System Controller registers are locked read to this register returns 0x00000000.

2.2.4 RUN profile register set

RUN profile register set contains the register group used to configure the device in RUN state. User needs to configure this group before applying run trigger by programming SYSC_TRGRUNCNTR to 0xAB. Applying invalid profile will be rejected by the device state logic and error flag is set. The validity of the profile can also be checked, before applying profile, by monitoring the bit SYSC_SYSSTSR:IRPARUN. This profile is also applied on device wake up from PSS. For invalid combinations refer to 2.3.2.8 RUN profile. This register set gets initialized on hard resets.

2.2.4.1 RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR)

This register is used to configure power domains in the RUN state. Power domain 2 and power domain 3 are always switched ON and cannot be configured in the RUN state.

RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR)

Figure 2-3. RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR)

SYSC_RUNPDCFGR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PD5ON	PD4ON	PD3ON	PD2ON
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R1WPS1	R1WPS1
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 2-4. RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	PD5ON	Power Domain 5 Configuration bit This bit controls power domain 5 switch ON and switch OFF. '0': Power domain 5 switch OFF request '1': Power domain 5 switch ON request
[2]	PD4ON	Power Domain 4 Configuration bit This bit controls power domain 4 switch ON and switch OFF. '0': Power domain 4 switch OFF request '1': Power domain 4 switch ON request
[1]	PD3ON	Power Domain 3 Configuration bit This bit controls power domain 3 switch ON and switch OFF. '0': Reserved '1': Power domain 3 switch ON request This bit is fixed to '1', in RUN profile i.e. power domain 3 is always switched ON when device is in RUN state. Writing '0' to this bit has no effect.

Table 2-4. RUN Profile Power Domain Configuration Register (SYSC_RUNPDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[0]	PD2ON	Power Domain 2 Configuration bit This bit controls power domain 2 switch ON and switch OFF. '0': Reserved '1': Power domain 2 switch ON request This bit is fixed to '1', in RUN profile i.e. power domain 2 is always switched ON when device is in RUN state. Writing '0' to this bit has no effect.

2.2.4.2 RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER)

This register is used to configure source clocks in the RUN state.

RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER)

Figure 2-4. RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER)

SYSC_RUNCKSRER															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXPLLEN	SSCGPLLEN	MAINPLLEN	SOSCEN	MOSCEN	RCOSCEN	SRCOSCEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R1WPS1	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Table 2-5. RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7]	read0	-
[6]	GFXPLLEN	GFX PLL Enable Control bit This bit is used to enable/disable GFX PLL oscillation circuit. '0': GFX PLL oscillator disable '1': GFX PLL oscillator enable Disabling of PLL oscillation circuit, when PLL oscillator clock is used in the device is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set). For PLL related settings refer to PLL interface chapter.
[5]	SSCGPLLEN	SSCG PLL Enable Control bit This bit is used to enable/disable SSCG PLL oscillation circuit. '0': SSCG PLL oscillator disable '1': SSCG PLL oscillator enable Disabling of PLL oscillation circuit, when PLL oscillator clock is used in the device is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set). For PLL related settings refer to PLL interface chapter.

Table 2-5. RUN Profile Clock Source Enable Register (SYSC_RUNCKSRER) bits

Bit Position	Bit Field Name	Bit Description
[4]	MAINPLLEN	Main PLL Enable Control bit This bit is used to enable/disable Main PLL oscillation circuit. '0': Main PLL oscillator disable '1': Main PLL oscillator enable Disabling of PLL oscillation circuit, when PLL oscillator clock is used in the device is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set). For PLL related settings refer to PLL interface chapter.
[3]	SOSCEN	Sub Oscillator Enable Control bit This bit is used to enable/disable Sub oscillation circuit. '0': Sub oscillator disable '1': Sub oscillator enable Disabling of Sub oscillation circuit, when Sub oscillator clock is used in the device is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).
[2]	MOSCEN	Main Oscillator Enable Control bit This bit is used to enable/disable Main oscillation circuit. '0': Main oscillator disable '1': Main oscillator enable Disabling of Main oscillation circuit, when Main oscillator clock is used in the device is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).
[1]	RCOSCEN	RC Oscillator Enable Control bit This bit is used to enable/disable RC oscillation circuit. '0': Reserved '1': RC oscillator enable In RUN profile this bit is fixed to '1' which means in RUN state RC clock is always enabled. Writing '0' on this bit has no effect.
[0]	SRCOSCEN	Slow RC Oscillator Enable Control bit This bit is used to enable/disable Slow RC oscillator circuit. '0': Slow RC oscillator disable '1': Slow RC oscillator enable Disabling of Slow RC oscillation circuit, when Slow RC oscillator clock is used in the device is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).

2.2.4.3 RUN Profile Clock Select Register (SYSC_RUNCKSELR)

This register is used to configure clock selection multiplexer for all clock domains in the RUN state.

RUN Profile Clock Select Register (SYSC_RUNCKSELR)

Figure 2-5. RUN Profile Clock Select Register (SYSC_RUNCKSELR)

SYSC_RUNCKSELR																															
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**
read0	read0	SPICSL[1]	SPICSL[0]	read0	read0	PIXCSL[1]	PIXCSL[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI3CSL[2]	PERI3CSL[1]	PERI3CSL[0]	read0	PERI1CSL[2]	PERI1CSL[1]	PERI1CSL[0]	read0	PERI0CSL[2]	PERI0CSL[1]	PERI0CSL[0]	read0	SYSCSL[2]	SYSCSL[1]	SYSCSL[0]
R0	R0	RWPS	RWPS	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	RWPS	RWPS	RWPS	R0	RWPS	RWPS	RWPS	R0	RWPS	RWPS	RWPS
0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-6. RUN Profile Clock Select Register (SYSC_RUNCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[31:30]	read0	-
[29:28]	SPICSL	SPI Clock Select for Graphics Subsystem These bits select clock for the Graphics Subsystem SPI clock. '00': SPI clock (Main PLL clock) is selected '01': External clock is selected '10' to '11': Clock is tied low
[27:26]	read0	-
[25:24]	PIXCSL	Pixel Clock Select for Graphics Subsystem These bits select clock for the Graphics Subsystem pixel clock. '00': GFX bus clock is selected '01': External clock is selected '10': GFX PLL clock is selected '11': Clock is tied low **Note:** It is recommended to tie clock to low, when Graphics Subsystem is in idle state.
[23:16]	read0	-
[15]	read0	-

Table 2-6. RUN Profile Clock Select Register (SYSC_RUNCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[14:12]	PERI3CSL	Peripheral Group 3 Clock Select Control bit These bits select the clock for the peripheral group 3. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low **Note:** It is recommended to tie clock to low, when all the peripherals connected to the peripheral group 3 are in idle state.
[11]	read0	-
[10:8]	PERI1CSL	Peripheral Group 1 Clock Select Control bit These bits select the clock for the peripheral group 1. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low **Note:** It is recommended to tie clock to low, when all the peripherals connected to the peripheral group 1 are in idle state.
[7]	read0	-
[6:4]	PERI0CSL	Peripheral Group 0 Clock Select Control bit These bits select the clock for the peripheral group 0. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low **Note:** It is recommended to tie clock to low, when all the peripherals connected to the peripheral group 0 are in idle state.
[3]	read0	-

Table 2-6. RUN Profile Clock Select Register (SYSC_RUNCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[2:0]	SYSCSL	System Clock Select Control bit These bits select the clock for the core group. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low User should select RC clock as system clock, if PD2 domain is switched OFF during PSS switching. Selecting other than RC clock in this scenario, sets SYSC_SYSSTSR:IRPAPSS bit. Selecting no clock in RUN state i.e. setting this field to '101' to '111' will generate invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).

2.2.4.4 RUN Profile Clock Enable Register (SYSC_RUNCKER)

This register is used to configure clock gating in the clock distribution logic in the RUN state.

RUN Profile Clock Enable Register (SYSC_RUNCKER)

Figure 2-6. RUN Profile Clock Enable Register (SYSC_RUNCKER)

SYSC_RUNCKER																															
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**
read0	ENSPIPD5	ENPIXPD5	ENGFXPD5	read0	read0	read0	ENCFGPD4	read0	read0	ENEXTBUSPD3	ENSPIPD3	ENMEMEPD3	ENHPMPD3	ENTRACEPD3	ENSYSPD3	read0	read0	read0	read0	ENPERI4PD2	ENPERI3PD2	read0	ENPERI1PD2	ENPERI0PD2	ENDMAPD2	ENTRACEPD2	ENHPMPD2	read0	read0	read0	ENCFGPD1
R0	RWPS	RWPS	RWPS	R0	R0	R0	RWPS	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R1WPS1	R0	R0	R0	R0	RWPS	RWPS	R0	RWPS	RWPS	RWPS	RWPS	R1WPS1	R0	R0	R0	R1WPS1
0	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	1

Table 2-7. RUN Profile Clock Enable Register (SYSC_RUNCKER) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30]	ENSPIPD5	Enable Graphics SPI Clock for Power Domain 5 This bit is used to enable/disable the SPI clock to the Graphics Sub-system. '0': SPI clock to the Graphics Subsystem is disabled '1': SPI clock to the Graphics Subsystem is enabled Setting this bit to '1' and setting power domain 5 config bit SYSC_RUNPDCFGR:PD5ON = '0' is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).
[29]	ENPIXPD5	Enable Graphics PLL Clock for Power Domain 5 This bit is used to enable/disable the pixel clock to the Graphics Sub-system. '0': Pixel clock to the Graphics Subsystem is disabled '1': Pixel clock to the Graphics Subsystem is enabled Setting this bit to '1' and setting power domain 5 config bit SYSC_RUNPDCFGR:PD5ON = '0' is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).

Table 2-7. RUN Profile Clock Enable Register (SYSC_RUNCKER) bits

Bit Position	Bit Field Name	Bit Description
[28]	ENGFXPD5	Enable Graphics Bus Clock for Power Domain 5 This bit is used to enable/disable bus clock to the Graphics Subsystem. '0': Bus clock to the Graphics Subsystem is disabled '1': Bus clock to the Graphics Subsystem is enabled Setting this bit to '1' and setting power domain 5 config bit SYSC_RUNPDCFGR:PD5ON = '0' is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).
[27:25]	read0	-
[24]	ENCFGPD4	Enable Configuration Clock for Power Domain 4 This bit is used to enable/disable clocks to the Retention RAM. '0': Retention RAM clock is disabled '1': Retention RAM clock is enabled Setting this bit to '1' and setting power domain 4 config bit SYSC_RUNPDCFGR:PD4ON = '0' is invalid setting. This generates invalid RUN profile error (SYSC_SYSSTSR:IRPARUN and SYSC_SYSSTSR:IRPAPSS bits are set).
[23:22]	read0	-
[21]	ENEXTBUSPD3	Enable External Bus Clock for Power Domain 3 This bit is used to enable/disable the clock to the external bus peripheral. '0': Clock to external bus peripheral disabled '1': Clock to external bus peripheral enabled
[20]	ENSPIPD3	Enable SPI Clock of HSSPI for Power Domain 3 This bit is used to enable/disable SPI clock to HSSPI module. '0': SPI clock of HSSPI is disabled '1': SPI clock of HSSPI is enabled By default, HSSPI uses its bus clock to generate SPI clock. This non-modulated clock is required for high frequency operation. For more details refer to the HSSPI specification.
[19]	ENMEMEPD3	Enable Memory External Clock for Power Domain 3 This bit is used to enable/disable the clock to the external memory group. '0': Clock to external memory group disabled '1': Clock to external memory group enabled
[18]	ENHPMPD3	Enable HPM Clock for Power Domain 3 This bit is used to enable/disable the clock to the AXI port of TCFLASH interface. '0': Clock to the AXI port of TCFLASH interface disabled '1': Clock to the AXI port of TCFLASH interface enabled

Table 2-7. RUN Profile Clock Enable Register (SYSC_RUNCKER) bits

Bit Position	Bit Field Name	Bit Description
[17]	ENTRACEPD3	Enable Trace Clock for Power Domain 3 This bit is used to enable/disable the clock to the debug and trace group in power domain 3. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled By default the clocks are enabled to the debug and trace group. User can switch OFF the debug and trace group. Disabling the trace clock, disables the debug and trace logic.
[16]	ENSYSPD3	Enable System Clock for Power Domain 3 This bit is used to enable/disable the clock to the core group. '0': Reserved '1': Clock to core group enabled In RUN profile this bit is fixed to '1' which means in RUN state core group clock is always enabled. **Note:** Since TCFLASH interface needs CLK_SYS, it has to be enabled even when Flash needs to be accessed by other master. If CPU is not required WFI instruction should be used, while keeping CLK_SYS enabled.
[15:12]	read0	-
[11]	ENPERI4PD2	Enable Peripheral Group 4 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 4. '0': Clock to peripheral group 4 disabled '1': Clock to peripheral group 4 enabled
[10]	ENPERI3PD2	Enable Peripheral Group 3 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 3. '0': Clock to peripheral group 3 disabled '1': Clock to peripheral group 3 enabled
[9]	read0	-
[8]	ENPERI1PD2	Enable Peripheral Group 1 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 1. '0': Clock to peripheral group 1 disabled '1': Clock to peripheral group 1 enabled
[7]	ENPERI0PD2	Enable Peripheral Group 0 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 0. '0': Clock to peripheral group 0 disabled '1': Clock to peripheral group 0 enabled
[6]	ENDMAPD2	Enable DMA Clock for Power Domain 2 This bit is used to enable/disable the clock to the DMA in power domain 2. '0': Clock to DMA disabled '1': Clock to DMA enabled

Table 2-7. RUN Profile Clock Enable Register (SYSC_RUNCKER) bits

Bit Position	Bit Field Name	Bit Description
[5]	ENTRACEPD2	Enable Trace Clock for Power Domain 2 This bit is used to enable/disable the clock to the debug and trace group in power domain 2. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled User can switch OFF the debug and trace group. Disabling the trace clock, disables the debug and trace logic.
[4]	ENHPMPD2	Enable HPM Clock for Power Domain 2 This bit is used to enable/disable the clock to the HPM bus matrix. '0': Reserved '1': Clock to HPM bus matrix enabled In RUN profile this bit is fixed to '1' which means in RUN state HPM bus matrix clock is always enabled.
[3:1]	read0	-
[0]	ENCFGPD1	Enable Configuration Clock for Power Domain 1 This bit is used to enable/disable the clock to the configuration group. '0': Reserved '1': Clock to configuration group enabled In RUN profile this bit is fixed to '1' which means in RUN state configuration group clock is always enabled.

2.2.4.5 RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0)

This register is used to configure clock division ratio for clock paths in the RUN state.

RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0)

Figure 2-7. RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0)

SYSC_RUNCKDIVR0																													
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**
read0	read0	read0	read0	read0	read0	CFGDIV[1]	CFGDIV[0]	read0	read0	read0	read0	read0	read0	HPMDIV[1]	HPMDIV[0]	read0	read0	read0	read0	read0	read0	TRACEDIV[1]	TRACEDIV[0]	read0	read0	read0	read0	SYSDIV[3]	SYSDIV[2]	SYSDIV[1]	SYSDIV[0]
R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-8. RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	CFGDIV	Configuration Clock Division Value These bits control the clock divider for the configuration group clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	HPMDIV	HPM Clock Division Value These bits control the clock divider for the HPM clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-

Table 2-8. RUN Profile Clock Divider Register 0 (SYSC_RUNCKDIVR0) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	TRACEDIV	Trace Clock Division Value These bits control the clock divider for the debug and trace clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[7:4]	read0	-
[3:0]	SYSDIV	System Clock Division Value These bits control the clock divider for the system clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.4.6 RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1)

This register is used to configure clock division ratio for clock paths in the RUN state.

RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1)

Figure 2-8. RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1)

SYSC_RUNCKDIVR1																													
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**
read0	read0	read0	read0	read0	read0	GFXDIV[1]	GFXDIV[0]	read0	read0	read0	read0	read0	read0	PERI4DIV[1]	PERI4DIV[0]	read0	read0	read0	read0	read0	read0	MEMEDIV[1]	MEMEDIV[0]	read0	read0	read0	read0	read0	EXTBUSDIV[2]	EXTBUSDIV[1]	EXTBUSDIV[0]
R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-9. RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	GFXDIV	Graphics Clock Division Value These bits control the clock divider for the Graphics Subsystem bus clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	PERI4DIV	Peripheral 4 Clock Division Value These bits control the clock divider for the peripheral group 4 clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-

Table 2-9. RUN Profile Clock Divider Register 1 (SYSC_RUNCKDIVR1) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	MEMEDIV	Memory External Clock Division Value These bits control the clock divider for the external memory group (HSSPI module) clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[7:3]	read0	-
[2:0]	EXTBUSDIV	External Bus Clock Division Value These bits control the clock divider for the external bus clock. For more details refer to Figure 2-132. '000': Clock division is bypassed (division by 1) '001': Divided by 2 '010': Divided by 4 '011': Divided by 8 ... '110': Divided by 64, '111': Divided by 128,

2.2.4.7 RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2)

This register is used to configure clock division ratio for clock paths in the RUN state.

RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2)

Figure 2-9. RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2)

SYSC_RUNCKDIVR2																															
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**
read0	read0	read0	read0	PERI3DIV[3]	PERI3DIV[2]	PERI3DIV[1]	PERI3DIV[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI1DIV[3]	PERI1DIV[2]	PERI1DIV[1]	PERI1DIV[0]	read0	read0	read0	read0	PERI0DIV[3]	PERI0DIV[2]	PERI0DIV[1]	PERI0DIV[0]
R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-10. RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:24]	PERI3DIV	Peripheral Group 3 Clock Division Value These bits control the clock divider for the peripheral group 3. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[23:16]	read0	-
[15:12]	read0	-
[11:8]	PERI1DIV	Peripheral Group 1 Clock Division Value These bits control the clock divider for the peripheral group 1. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:4]	read0	-

Table 2-10. RUN Profile Clock Divider Register 2 (SYSC_RUNCKDIVR2) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	PERI0DIV	Peripheral Group 0 Clock Division Value These bits control the clock divider for the peripheral group 0. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.4.8 RUN Profile PLL Control Register (SYSC_RUNPLLCNTR)

This register is used to configure Main PLL in the RUN state. For PLL stabilization time settings refer to 2.3.5.3 Source clocks.

RUN Profile PLL Control Register (SYSC_RUNPLLCNTR)

Figure 2-10. RUN Profile PLL Control Register (SYSC_RUNPLLCNTR)

SYSC_RUNPLLCNTR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	PLLDIVN[6]	PLLDIVN[5]	PLLDIVN[4]	PLLDIVN[3]	PLLDIVN[2]	PLLDIVN[1]	PLLDIVN[0]	read0	read0	read0	read0	PLLDIVM[3]	PLLDIVM[2]	PLLDIVM[1]	PLLDIVM[0]	read0	read0	read0	read0	read0	read0	PLLDIVL[1]	PLLDIVL[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-11. RUN Profile PLL Control Register (SYSC_RUNPLLCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	read0	-
[22:16]	PLLDIVN	PLL Multiplication-by-N Value These bits control the multiplication value for Main PLL clock. For the PLL programming refer to Main PLL interface. '0000000': PLL input clock is multiplied by 1 '0000001': PLL input clock is multiplied by 2 ... '1111111': PLL input clock is multiplied by 128 Refer to device specific datasheet for Main PLL output frequency range. Do not program the value which falls outside the range. User should program multiplication factor accordingly.
[15:12]	read0	-
[11:8]	PLLDIVM	PLL Division-by-M Value These bits control the clock divider for Main PLL output clock. For more details refer to Figure 2-132. '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:2]	read0	-

Table 2-11. RUN Profile PLL Control Register (SYSC_RUNPLLCNTR) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	PLLDIVL	PLL Input Division Value These bits control the input clock divider for the Main PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.4.9 RUN Profile SSCG Control Register 0 (SYSC_RUNSSCGCNTR0)

This register is used to configure SSCG PLL in the RUN state. For SSCG-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

RUN Profile SSCG Control Register 0 (SYSC_RUNSSCGCNTR0)

Figure 2-11. RUN Profile SSCG Control Register 0 (SYSC_RUNSSCGCNTR0)

SYSC_RUNSSCGCNTR0																															
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**
read0	read0	read0	SSCGDIVP[4]	SSCGDIVP[3]	SSCGDIVP[2]	SSCGDIVP[1]	SSCGDIVP[0]	read0	read0	SSCGDIVN[5]	SSCGDIVN[4]	SSCGDIVN[3]	SSCGDIVN[2]	SSCGDIVN[1]	SSCGDIVN[0]	read0	read0	read0	read0	SSCGDIVM[3]	SSCGDIVM[2]	SSCGDIVM[1]	SSCGDIVM[0]	read0	read0	read0	read0	read0	read0	SSCGDIVL[1]	SSCGDIVL[0]
R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-12. RUN Profile SSCG ControlRegister 0 (SYSC_RUNSSCGCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	SSCGDIVP	SSCG PLL Multiplication-by-P Value These bits along with SYSC_RUNSSCGCNTR0:SSCGDIVN bits define the multiplication value for SSCG PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_RUNSSCGCN-TR0:SSCGDIVN x 1) '00010':PLL input clock multiplied by (SYSC_RUNSSCGCN-TR0:SSCGDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_RUNSSCGCN-TR0:SSCGDIVN x 31) Refer to device specific datasheet for SSCG PLL output frequency range. User should program multiplication factor accordingly.
[23:22]	read0	-

Table 2-12. RUN Profile SSCG ControlRegister 0 (SYSC_RUNSSCGCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	SSCGDIVN	SSCG PLL Multiplication-by-N Value These bits along with SYSC_RUNSSCGCNTR0:SCGDIVP bits define the multiplication value for SSCG PLL clock. For SSCG PLL programming refer to SSCG and GFX PLL interface. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_RUNSSCGCN-TR0:SSCGDIVP x 2) '000011': PLL input clock multiplied by (SYSC_RUNSSCGCN-TR0:SSCGDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_RUNSSCGCN-TR0:SSCGDIVP x 63) Refer to device specific datasheet for SSCG PLL output frequency range. User should program multiplication factor accordingly.
[15:12]	read0	-
[11:8]	SSCGDIVM	SSCG PLL Division-by-M Value These bits control the clock divider for the division value for SSCG PLL. For more details refer to Figure 2-132. '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:2]	read0	-
[1:0]	SSCGDIVL	SSCG PLL Input Division Value These bits control the input clock divider for the SSCG PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.4.10 RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1)

This register is used to configure SSCG PLL in the RUN state.

RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1)

Figure 2-12. RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1)

SYSC_RUNSSCGCNTR1																													
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**
read0	read0	read0	read0	read0	read0	read0	SSCGSSEN	read0	read0	read0	read0	read0	SSCGFREQ[1]	SSCGFREQ[0]	SSCGMODE	read0	read0	read0	read0	read0	read0	SSCGRATE[9]	SSCGRATE[8]	SSCGRATE[7]	SSCGRATE[6]	SSCGRATE[5]	SSCGRATE[4]	SSCGRATE[3]	SSCGRATE[2]	SSCGRATE[1]	SSCGRATE[0]
R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-13. RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	SSCGSSEN	SSCG PLL Modulation Enable Control These bits enable/disable the modulation for SSCG PLL. '0': SSCG PLL modulation is disabled '1': SSCG PLL modulation is enabled
[23:19]	read0	-
[18:17]	SSCGFREQ	SSCG PLL Modulation Frequency Select These bits select the modulation frequency rate for SSCG PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	SSCGMODE	SSCG PLL Modulator Mode This bit controls the modulation mode of SSCG PLL. '0': SSCG PLL works in down spread mode '1': SSCG PLL works in center spread mode
[15:10]	read0	-

Table 2-13. RUN Profile SSCG Control Register 1 (SYSC_RUNSSCGCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	SSCGRATE	SSCG PLL Modulation Rate Control Value These bits control the modulation rate for SSCG PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.4.11 RUN Profile Graphics PLL Control Register 0 (SYSC_RUNGFXCNTR0)

This register is used to configure Graphics PLL in the RUN state. For GFX-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

RUN Profile Graphics PLL Control Register 0 (SYSC_RUNGFXCNTR0)

Figure 2-13. RUN Profile GFX PLL Control Register 0 (SYSC_RUNGFXCNTR0)

SYSC_RUNGFXCNTR0																															
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**
read0	read0	read0	GFXDIVP[4]	GFXDIVP[3]	GFXDIVP[2]	GFXDIVP[1]	GFXDIVP[0]	read0	read0	GFXDIVN[5]	GFXDIVN[4]	GFXDIVN[3]	GFXDIVN[2]	GFXDIVN[1]	GFXDIVN[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXDIVL[1]	GFXDIVL[0]
R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-14. RUN Profile GFX PLL Control Register 0 (SYSC_RUNGFXCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	GFXDIVP	Graphics PLL Division-by-P Value These bits along with SYSC_RUNGFXCNTR0:GFXDIVN bits define the multiplication value for GFX PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_RUNGFXCN-TR0:GFXDIVN x 1) '00010':PLL input clock multiplied by (SYSC_RUNGFXCN-TR0:GFXDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_RUNGFXCN-TR0:GFXDIVN x 31) Refer to device specific datasheet for GFX PLL output frequency range. User should program multiplication factor accordingly.
[23:22]	read0	-

Table 2-14. RUN Profile GFX PLL Control Register 0 (SYSC_RUNGFXCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	GFXDIVN	Graphics PLL Division-by-N Value These bits along with SYSC_RUNGFXCNTR0:GFXDIVP bits define the multiplication value for GFX PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_RUNGFXCN-TR0:GFXDIVP x 2) '000011': PLL input clock multiplied by (SYSC_RUNGFXCN-TR0:GFXDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_RUNGFXCN-TR0:GFXDIVP x 63) Refer to device specific datasheet for GFX PLL output frequency range. User should program multiplication factor accordingly.
[15:8]	read0	-
[7:2]	read0	-
[1:0]	GFXDIVL	Graphics PLL Input Division Value These bits control the input clock divider for the GFX PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.4.12 RUN Profile Graphics PLL Control Register 1 (SYSC_RUNGFXCNTR1)

This register is used to configure Graphics PLL in the RUN state.

RUN Profile Graphics PLL Control Register 1 (SYSC_RUNGFXCNTR1)

Figure 2-14. RUN Profile GFX PLL Control Register 1 (SYSC_RUNGFXCNTR1)

SYSC_RUNGFXCNTR1																													
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**
read0	read0	read0	read0	read0	read0	read0	GFXSSEN	read0	read0	read0	read0	read0	GFXFREQ[1]	GFXFREQ[0]	GFXMODE	read0	read0	read0	read0	read0	read0	GFXRATE[9]	GFXRATE[8]	GFXRATE[7]	GFXRATE[6]	GFXRATE[5]	GFXRATE[4]	GFXRATE[3]	GFXRATE[2]	GFXRATE[1]	GFXRATE[0]
R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-15. RUN Profile GFX PLL Control Register 1 (SYSC_RUNGFXCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	GFXSSEN	GFX PLL Modulation Enable Control These bits enable/disable the modulation for GFX PLL. '0': GFX PLL modulation is disabled '1': GFX PLL modulation is enabled
[23:19]	read0	-
[18:17]	GFXFREQ	Graphics PLL Modulation Frequency Select These bits select the modulation frequency rate for GFX PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	GFXMODE	Graphics PLL Modulator Mode This bit controls the modulation mode of GFX PLL. '0': GFX PLL works in down spread mode '1': GFX PLL works in center spread mode
[15:10]	read0	-

Table 2-15. RUN Profile GFX PLL Control Register 1 (SYSC_RUNGFXCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	GFXRATE	Graphics PLL Modulation Rate Control Value These bits control the modulation rate for GFX PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.4.13 RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR)

This register is used to configure LVD in the RUN state.

RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR)

Figure 2-15. RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR)

SYSC_RUNLVDCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SV12[2]	SV12[1]	SV12[0]	LVDR12	LVDE12	read0	read0	read0	SV33[2]	SV33[1]	SV33[0]	LVDR33	LVDE33	read0	read0	read0	SV50[2]	SV50[1]	SV50[0]	LVDR50	LVDE50
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1

Table 2-16. RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:21]	read0	-
[20:18]	SV12	Reference Voltage Selection for 1.2 V LVD These bits control analog threshold for Low Voltage Detection on 1.2 V core supply. '000': 0.5 V '001': 0.6 V '011': 0.7 V '010': 0.8 V '110': 0.9 V '111': 1.0 V '101': 1.1 V '100': 1.2 V On reset, 0.8 V is chosen as reference voltage.
[17]	LVDR12	Low Voltage Detector Reset/Interrupt Select for 1.2 V This bit is used to enable interrupt/reset for LVD on 1.2 V core supply. '0': Low Voltage Detect on 1.2 V sets interrupt flag SYSC_SYSERRR:LVD12IF '1': Low Voltage Detect on 1.2 V generates reset

Table 2-16. RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[16]	LVDE12	Low Voltage Detector Enable for 1.2 V This bit is used to enable/disable LVD functionality on 1.2 V core supply. '0': LVD on 1.2 V disabled '1': LVD on 1.2 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.
[15:13]	read0	-
[12:10]	SV33	Reference Voltage Selection for 3.3 V LVD These bits control analog threshold for Low Voltage Detection on 3.3 V graphics IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V Others: Reserved User should not configure reserved values to avoid unexpected reset. On reset, 2.6 V is chosen as reference voltage.
[9]	LVDR33	Low Voltage Detector Reset/Interrupt Select for 3.3 V This bit is used to enable interrupt/reset for LVD on 3.3 V graphics IO supply. '0': Low Voltage Detect on 3.3 V sets interrupt flag SYSC_SYSERRR:LVD33IF '1': Low Voltage Detect on 3.3 V generates reset
[8]	LVDE33	Low Voltage Detector Enable for 3.3 V This bit is used to enable/disable LVD functionality on 3.3 V graphics IO supply. '0': LVD on 3.3 V disabled '1': LVD on 3.3 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.
[7:5]	read0	-
[4:2]	SV50	Reference Voltage Selection for 5.0 V LVD These bits control analog threshold for Low Voltage Detection on 5.0 V IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V '110': 3.7 V '111': 3.9 V '101': 4.1 V '100': 4.3 V On reset, 2.6 V is chosen as reference voltage.

Table 2-16. RUN Profile Low Voltage Detector Configuration Register (SYSC_RUNLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[1]	LVDR50	Low Voltage Detector Reset/Interrupt Select for 5.0 V This bit is used to enable interrupt/reset for LVD on 5.0 V IO supply. '0': Low Voltage Detect on 5.0 V sets interrupt flag SYSC_SYSERRR:LVD50IF '1': Low Voltage Detect on 5.0 V generates reset
[0]	LVDE50	Low Voltage Detector Enable for 5.0 V This bit is used to enable/disable LVD functionality on 5.0 V IO supply. '0': LVD on 5.0 V disabled '1': LVD on 5.0 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.

2.2.4.14 RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR)

This register is used to configure CSV in the RUN state.

RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR)

Figure 2-16. RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR)

SYSC_RUNCSVCFGR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GPCSVE	SPCSVE	MPCSVE	SOCSVE	MOCSVE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-17. RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:5]	read0	-
[4]	GPCSVE	Graphics PLL Clock Supervisor Enable This bit enables clock supervision on Graphics PLL clock. '0': Graphics PLL clock supervision disabled '1': Graphics PLL clock supervision enabled Clock and reference clock (See 2.3.7 Clock Supervisor) have to be enabled to enable supervision. once supervision has been enabled it can only be disabled when the clock is also disabled. Device state logic rejects these settings and SYSC_SYSSTSR:IRPARUN/ SYSC_SYSSTSR:IRPAPSS error flags are set.
[3]	SPCSVE	SSCG PLL Clock Supervisor Enable This bit enables clock supervision on SSCG PLL clock. '0': SSCG PLL clock supervision disabled '1': SSCG PLL clock supervision enabled Clock and reference clock (See 2.3.7 Clock Supervisor) have to be enabled to enable supervision. once supervision has been enabled it can only be disabled when the clock is also disabled. Device state logic rejects these settings and SYSC_SYSSTSR:IRPARUN/ SYSC_SYSSTSR:IRPAPSS error flags are set.

Table 2-17. RUN Profile Clock Supervisor Configuration Register (SYSC_RUNCSVCFGR) bits

Bit Position	Bit Field Name	Bit Description
[2]	MPCSVE	Main PLL Clock Supervisor Enable This bit enables clock supervision on Main PLL clock. '0': Main PLL clock supervision disabled '1': Main PLL clock supervision enabled Clock and reference clock (See 2.3.7 Clock Supervisor) have to be enabled to enable supervision. once supervision has been enabled it can only be disabled when the clock is also disabled. Device state logic rejects these settings and SYSC_SYSSTSR:IRPARUN/ SYSC_SYSSTSR:IRPAPSS error flags are set.
[1]	SOCSVE	Sub Oscillator Clock Supervisor Enable This bit enables clock supervision on Sub clock. '0': Sub clock supervision disabled '1': Sub clock supervision enabled Clock and reference clock (See 2.3.7 Clock Supervisor) have to be enabled to enable supervision. once supervision has been enabled it can only be disabled when the clock is also disabled. Device state logic rejects these settings and SYSC_SYSSTSR:IRPARUN/ SYSC_SYSSTSR:IRPAPSS error flags are set.
[0]	MOCSVE	Main Oscillator Clock Supervisor Enable This bit enables clock supervision on Main clock. '0': Main clock supervision disabled '1': Main clock supervision enabled Clock and reference clock (See 2.3.7 Clock Supervisor) have to be enabled to enable supervision. once supervision has been enabled it can only be disabled when the clock is also disabled. Device state logic rejects these settings and SYSC_SYSSTSR:IRPARUN/ SYSC_SYSSTSR:IRPAPSS error flags are set.

2.2.4.15 Trigger RUN Control Register (SYSC_TRGRUNCNTR)

This register is used to apply new RUN profile settings. Device state logic gets triggered when 0xAB is written into this register and new RUN profile settings are applied.

Trigger RUN Control Register (SYSC_TRGRUNCNTR)

Figure 2-17. Trigger RUN Control Register (SYSC_TRGRUNCNTR)

SYSC_TRGRUNCNTR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	APPLYRUN[7]	APPLYRUN[6]	APPLYRUN[5]	APPLYRUN[4]	APPLYRUN[3]	APPLYRUN[2]	APPLYRUN[1]	APPLYRUN[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-18. Trigger RUN Control Register (SYSC_TRGRUNCNTR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:0]	APPLYRUN	Apply RUN Trigger Register bits These bits when configured as 0xAB, trigger the device state logic to apply new RUN settings. This register is cleared automatically by hardware. These bits need to be configured only after RUN profile is completely configured. Writing value other than 0xAB will set SYSC_SYSERRR:RUNTRGEIF. Once the System Controller state switching starts i.e. when SYSC_SYSSTSR:RUNBUSY = '1', writing any value to this register sets SYSC_SYSERRR:TRGERRIF register bit. Reading this register always returns '0'.

2.2.5 PSS profile register set

PSS profile register set contains the register group used to configure the device in PSS. User needs to configure this group along with RUN profile (for wakeup) before PSS state switching by executing WFI and programming SYSC_PSSENR:PSSEN to 0xBA. Applying PSS/RUN invalid profile will be rejected by the device state logic and error flag is set. The validity of the PSS profile can also be checked before applying profile by monitoring the bit SYSC_SYSSTSR:IPPAPSS. Similarly the validity of RUN profile required for wakeup can be monitored by SYSC_SYSSTSR:IRPAPSS bit. For invalid combinations refer to 2.3.2.9 PSS profile and for details on profile error reset refer to 2.3.3 Reset and power up. These registers set get initialized on hard resets.

2.2.5.1 PSS Profile Power Domain Configuration Register (SYSC_PSSPDCFGR)

This register is used to configure power domains in the PSS.

PSS Profile Power Domain Configuration Register (SYSC_PSSPDCFGR)

Figure 2-18. PSS Profile Power Domain Configuration Register (SYSC_PSSPDCFGR)

SYSC_PSSPDCFGR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PD5ON	PD4ON	PD3ON	PD2ON
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 2-19. PSS Profile Power Domain Configuration Register (SYSC_PSSPDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	PD5ON	Power Domain 5 Configuration bit This bit controls power domain 5 switch ON and switch OFF. '0': Power domain 5 switch OFF request '1': Power domain 5 switch ON request
[2]	PD4ON	Power Domain 4 Configuration bit This bit controls power domain 4 switch ON and switch OFF. '0': Power domain 4 switch OFF request '1': Power domain 4 switch ON request
[1]	PD3ON	Power Domain 3 Configuration bit This bit controls power domain 3 switch ON and switch OFF. '0': Power domain 3 switch OFF request '1': Power domain 3 switch ON request
[0]	PD2ON	Power Domain 2 Configuration bit This bit controls power domain 2 switch ON and switch OFF. '0': Power domain 2 switch OFF request '1': Power domain 2 switch ON request The power domain 2 can be switched OFF only when power domain 3 and power domain 5 are switched OFF. Configuration of power domain 5 or power domain 3 to ON state when power domain 2 is OFF (SYSC_PDSTSR:PD2ON = '0') sets SYSC_SYSSTSR:IPPAPSS.

2.2.5.2 PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER)

This register is used to configure source clocks in the PSS.

PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER)

Figure 2-19. PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER)

SYSC_PSSCKSRER															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXPLLEN	SSCGPLLEN	MAINPLLEN	SOSCEN	MOSCEN	RCOSCEN	SRCOSCEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Table 2-20. PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7]	read0	-
[6]	GFXPLLEN	GFX PLL Enable Control bit This bit is used to enable/disable GFX PLL oscillation circuit. '0': GFX PLL oscillator disable '1': GFX PLL oscillator enable Disabling of PLL oscillation circuit, when PLL oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set). For PLL related settings refer to PLL interface chapter.
[5]	SSCGPLLEN	SSCG PLL Enable Control bit This bit is used to enable/disable SSCG PLL oscillation circuit. '0': SSCG PLL oscillator disable '1': SSCG PLL oscillator enable Disabling of PLL oscillation circuit, when PLL oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set). For PLL related settings refer to PLL interface chapter.

Table 2-20. PSS Profile Clock Source Enable Register (SYSC_PSSCKSRER) bits

Bit Position	Bit Field Name	Bit Description
[4]	MAINPLLEN	Main PLL Enable Control bit This bit is used to enable/disable Main PLL oscillation circuit. '0': Main PLL oscillator disable '1': Main PLL oscillator enable Disabling of PLL oscillation circuit, when PLL oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set). For PLL related settings refer to PLL interface chapter.
[3]	SOSCEN	Sub Oscillator Enable Control bit This bit is used to enable/disable Sub oscillation circuit. '0': Sub oscillator disable '1': Sub oscillator enable Disabling of sub oscillation circuit, when Sub oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[2]	MOSCEN	Main Oscillator Enable Control bit This bit is used to enable/disable Main oscillation circuit. '0': Main oscillator disable '1': Main oscillator enable Disabling of Main oscillation circuit, when Main oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[1]	RCOSCEN	RC Oscillator Enable Control bit This bit is used to enable/disable RC oscillation circuit. '0': RC oscillator disable '1': RC oscillator enable Disabling of RC oscillation circuit, when RC oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[0]	SRCOSCEN	Slow RC Oscillator Enable Control bit This bit is used to enable/disable Slow RC oscillator circuit. '0': Slow RC oscillator disable '1': Slow RC oscillator enable Disabling of Slow RC oscillation circuit, when Slow RC oscillator clock is used in the device is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).

2.2.5.3 PSS Profile Clock Select Register (SYSC_PSSCKSELR)

This register controls the clock selection multiplexer for all clock domains in the PSS.

PSS Profile Clock Select Register (SYSC_PSSCKSELR)

Figure 2-20. PSS Profile Clock Select Register (SYSC_PSSCKSELR)

SYSC_PSSCKSELR																														
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**
read0	read0	SPICSL[1]	SPICSL[0]	read0	read0	PIXCSL[1]	PIXCSL[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI3CSL[2]	PERI3CSL[1]	PERI3CSL[0]	read0	PERI1CSL[2]	PERI1CSL[1]	PERI1CSL[0]	read0	PERI0CSL[2]	PERI0CSL[1]	PERI0CSL[0]	read0	SYSCSL[2]	SYSCSL[1]	SYSCSL[0]
R0	R0	RWPS	RWPS	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	RWPS	RWPS	RWPS	R0	RWPS	RWPS	RWPS	R0	RWPS	RWPS	RWPS
0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-21. PSS Profile Clock Select Register (SYSC_PSSCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[31:30]	read0	-
[29:28]	SPICSL	SPI Clock Select for Graphics Subsystem These bits selects clock for the Graphics Subsystem SPI clock. '00': SPI clock (Main PLL clock) is selected '01': External clock is selected '10' to '11': Clock is tied low
[27:26]	read0	-
[25:24]	PIXCSL	Pixel Clock Select for Graphics Subsystem These bits select clock for the Graphics Subsystem pixel clock. '00': GFX bus clock is selected '01': External clock is selected '10': GFX PLL clock is selected '11': Clock is tied low **Note:** It is recommended to tie clock to low, when Graphics Subsystem is in idle state.
[23:16]	read0	-
[15]	read0	-

Table 2-21. PSS Profile Clock Select Register (SYSC_PSSCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[14:12]	PERI3CSL	Peripheral Group 3 Clock Select Control bit These bits select the clock for the peripheral group 3. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low **Note:** It is recommended to tie clock to low, when all the peripherals connected to the peripheral group 3 are in idle state.
[11]	read0	-
[10:8]	PERI1CSL	Peripheral Group 1 Clock Select Control bit These bits select the clock for the peripheral group 1. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low **Note:** It is recommended to tie clock to low, when all the peripherals connected to the peripheral group 1 are in idle state.
[7]	read0	-
[6:4]	PERI0CSL	Peripheral Group 0 Clock Select Control bit These bits select the clock for the peripheral group 0. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low It is recommended to tie clock to low, when all the peripherals connected to the peripheral group 0 are in idle state.
[3]	read0	-
[2:0]	SYSCSL	System Clock Select Control bit These bits select the clock for the core group. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low

2.2.5.4 PSS Profile Clock Enable Register (SYSC_PSSCKER)

This register controls the clock gating in the clock distribution logic in the PSS.

PSS Profile Clock Enable Register (SYSC_PSSCKER)

Figure 2-21. PSS Profile Clock Enable Register (SYSC_PSSCKER)

SYSC_PSSCKER																													
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**
read0	ENSPIPD5	ENPIXPD5	ENGFXPD5	read0	read0	read0	ENCFGPD4	read0	read0	ENEXTBUSPD3	ENSPIPD3	ENMEMEPD3	ENHPMPD3	ENTRACEPD3	ENSYSPD3	read0	read0	read0	read0	ENPERI4PD2	ENPERI3PD2	read0	ENPERI1PD2	ENPERI0PD2	ENDMAPD2	ENTRACEPD2	ENHPMPD2	read0	read0	read0	ENCFGPD1
R0	RWPS	RWPS	RWPS	R0	R0	R0	RWPS	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0WPS0	R0	R0	R0	R0	RWPS	RWPS	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	RWPS
0	1	1	1	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	1

Table 2-22. PSS Profile Clock Enable Register (SYSC_PSSCKER) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30]	ENSPIPD5	Enable Graphics SPI Clock for Power Domain 5 This bit is used to enable/disable the SPI clock to the Graphics Sub-system. '0': SPI Clock to the Graphics Subsystem is disabled '1': SPI Clock to the Graphics Subsystem is enabled Setting this bit to '1' and setting power domain 5 config bit SYSC_PSSPDCFGR:PD5ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[29]	ENPIXPD5	Enable Graphics PLL Clock for Power Domain 5 This bit is used to enable/disable the pixel clock to the Graphics Sub-system. '0': Pixel clock to the Graphics Subsystem is disabled '1': Pixel clock to the Graphics Subsystem is enabled Setting this bit to '1' and setting power domain 5 config bit SYSC_PSSPDCFGR:PD5ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).

Table 2-22. PSS Profile Clock Enable Register (SYSC_PSSCKER) bits

Bit Position	Bit Field Name	Bit Description
[28]	ENGFXPD5	Enable Graphics Bus Clock for Power Domain 5 This bit is used to enable/disable bus clock to the Graphics Subsystem. '0': Bus clock to the Graphics Subsystem is disabled '1': Bus clock to the Graphics Subsystem is enabled Setting this bit to '1' and setting power domain 5 config bit SYSC_PSSPDCFGR:PD5ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[27:25]	read0	-
[24]	ENCFGPD4	Enable Configuration Clock for Power Domain 4 This bit is used to enable/disable clocks to the Retention RAM. '0': Retention RAM clock is disabled '1': Retention RAM clock is enabled Setting this bit to '1' and setting power domain 4 config bit SYSC_PSSPDCFGR:PD4ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[23:22]	read0	-
[21]	ENEXTBUSPD3	Enable External Bus Clock for Power Domain 3 This bit is used to enable/disable the clock to the external bus peripheral. '0': Clock to external bus peripheral disabled '1': Clock to external bus peripheral enabled Setting this bit to '1' and setting power domain 3 config bit SYSC_PSSPDCFGR:PD3ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[20]	ENSPIPD3	Enable SPI Clock of HSSPI for Power Domain 3 This bit is used to enable/disable SPI clock to HSSPI module. '0': SPI clock of HSSPI is disabled '1': SPI clock of HSSPI is enabled By default, HSSPI uses its bus clock to generate SPI clock. This non-modulated clock is required for high frequency operation. For more details refer to the HSSPI specification. Setting this bit to '1' and setting power domain 3 config bit SYSC_PSSPDCFGR:PD3ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).

Table 2-22. PSS Profile Clock Enable Register (SYSC_PSSCKER) bits

Bit Position	Bit Field Name	Bit Description
[19]	ENMEMEPD3	Enable Memory External Clock for Power Domain 3 This bit is used to enable/disable the clock to the external memory group. '0': Clock to external memory group disabled '1': Clock to external memory group enabled Setting this bit to '1' and setting power domain 3 config bit SYSC_PSSPDCFGR:PD3ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[18]	ENHPMPD3	Enable HPM Clock for Power Domain 3 This bit is used to enable/disable the clock to the AXI port of TCFLASH interface. '0': Clock to the AXI port of TCFLASH interface disabled '1': Clock to the AXI port of TCFLASH interface enabled Setting this bit to '1' and setting power domain 3 config bit SYSC_PSSPDCFGR:PD3ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[17]	ENTRACEPD3	Enable Trace Clock for Power Domain 3 This bit is used to enable/disable the clock to the debug and trace group in power domain 3. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled By default the clocks are enabled to the debug and trace group. User can switch OFF the debug and trace group. Disabling the trace clock, disables the debug and trace logic. Setting this bit to '1' and setting power domain 3 config bit SYSC_PSSPDCFGR:PD3ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[16]	ENSYSPD3	Enable System Clock for Power Domain 3 This bit is used to enable/disable the clock to the core group. '0': Clock to core group disabled '1': Reserved In PSS, system clock is always gated or cut. Setting this bit to '1' and setting power domain 3 config bit SYSC_PSSPDCFGR:PD3ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[15:12]	read0	-

Table 2-22. PSS Profile Clock Enable Register (SYSC_PSSCKER) bits

Bit Position	Bit Field Name	Bit Description
[11]	ENPERI4PD2	Enable Peripheral Group 4 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 4. '0': Clock to peripheral group 4 disabled '1': Clock to peripheral group 4 enabled Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[10]	ENPERI3PD2	Enable Peripheral Group 3 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 3. '0': Clock to peripheral group 3 disabled '1': Clock to peripheral group 3 enabled Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[9]	read0	-
[8]	ENPERI1PD2	Enable Peripheral Group 1 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 1. '0': Clock to peripheral group 1 disabled '1': Clock to peripheral group 1 enabled Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[7]	ENPERI0PD2	Enable Peripheral Group 0 Clock for Power Domain 2 This bit is used to enable/disable the clock to the peripheral group 0. '0': Clock to peripheral group 0 disabled '1': Clock to peripheral group 0 enabled Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[6]	ENDMAPD2	Enable DMA Clock for Power Domain 2 This bit is used to enable/disable the clock to the DMA in power domain 2. '0': Clock to DMA disabled '1': Clock to DMA enabled Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).

Table 2-22. PSS Profile Clock Enable Register (SYSC_PSSCKER) bits

Bit Position	Bit Field Name	Bit Description
[5]	ENTRACEPD2	Enable Trace Clock for Power Domain 2 This bit is used to enable/disable the clock to the debug and trace group in power domain 2. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled User can switch OFF the debug and trace group. Disabling the trace clock, disables the debug and trace logic. Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[4]	ENHPMPD2	Enable HPM Clock for Power Domain 2 This bit is used to enable/disable the clock to the HPM bus matrix. '0': Clock to HPM bus matrix disabled '1': Clock to HPM bus matrix enabled Setting this bit to '1' and setting power domain 2 config bit SYSC_PSSPDCFGR:PD2ON = '0' is invalid setting. This generates invalid PSS profile error (SYSC_SYSSTSR:IPPAPSS bit is set).
[3:1]	read0	-
[0]	ENCFGPD1	Enable Configuration Clock for Power Domain 1 This bit is used to enable/disable the clock to the configuration group. '0': Clock to configuration group disabled '1': Clock to configuration group enabled

2.2.5.5 PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0)

This register controls the clock division ratio for clock paths in the PSS.

PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0)

Figure 2-22. PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0)

SYSC_PSSCKDIVR0																															
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**
read0	read0	read0	read0	read0	read0	CFGDIV[1]	CFGDIV[0]	read0	read0	read0	read0	read0	read0	HPMDIV[1]	HPMDIV[0]	read0	read0	read0	read0	read0	read0	TRACEDIV[1]	TRACEDIV[0]	read0	read0	read0	read0	SYSDIV[3]	SYSDIV[2]	SYSDIV[1]	SYSDIV[0]
R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-23. PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	CFGDIV	Configuration Clock Division Value These bits control the clock divider for the configuration group clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	HPMDIV	HPM Clock Division Value These bits control the clock divider for the HPM clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-

Table 2-23. PSS Profile Clock Divider Register 0 (SYSC_PSSCKDIVR0) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	TRACEDIV	Trace Clock Division Value These bits control the clock divider for the debug and trace clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[7:4]	read0	-
[3:0]	SYSDIV	System Clock Division Value These bits control the clock divider for the system clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.5.6 PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1)

This register controls the clock division ratio for clock paths in the PSS.

PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1)

Figure 2-23. PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1)

SYSC_PSSCKDIVR1																															
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**
read0	read0	read0	read0	read0	read0	GFXDIV[1]	GFXDIV[0]	read0	read0	read0	read0	read0	read0	PERI4DIV[1]	PERI4DIV[0]	read0	read0	read0	read0	read0	read0	MEMEDIV[1]	MEMEDIV[0]	read0	read0	read0	read0	read0	EXTBUSDIV[2]	EXTBUSDIV[1]	EXTBUSDIV[0]
R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-24. PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	GFXDIV	Graphics Clock Division Value These bits control the clock divider for the Graphics Subsystem bus clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	PERI4DIV	Peripheral 4 Clock Division Value These bits control the clock divider for the peripheral group 4 clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-

Table 2-24. PSS Profile Clock Divider Register 1 (SYSC_PSSCKDIVR1) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	MEMEDIV	Memory External Clock Division Value These bits control the clock divider for the external memory group (HSSPI module) clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[7:3]	read0	-
[2:0]	EXTBUSDIV	External Bus Clock Division Value These bits control the clock divider for the External bus clock. For more details refer to Figure 2-132. '000': Clock division is bypassed (division by 1) '001': Divided by 2 '010': Divided by 4 '011': Divided by 8 ... '110': Divided by 64, '111': Divided by 128,

2.2.5.7 PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2)

This register controls the clock division ratio for clock paths in the PSS.

PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2)

Figure 2-24. PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2)

SYSC_PSSCKDIVR2																															
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**
read0	read0	read0	read0	PERI3DIV[3]	PERI3DIV[2]	PERI3DIV[1]	PERI3DIV[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI1DIV[3]	PERI1DIV[2]	PERI1DIV[1]	PERI1DIV[0]	read0	read0	read0	read0	PERI0DIV[3]	PERI0DIV[2]	PERI0DIV[1]	PERI0DIV[0]
R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-25. PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:24]	PERI3DIV	Peripheral Group 3 Clock Division Value These bits control the clock divider for the peripheral group 3. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[23:16]	read0	-
[15:12]	read0	-
[11:8]	PERI1DIV	Peripheral Group 1 Clock Division Value These bits control the clock divider for the peripheral group 1. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:4]	read0	-

Table 2-25. PSS Profile Clock Divider Register 2 (SYSC_PSSCKDIVR2) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	PERI0DIV	Peripheral Group 0 Clock Division Value These bits control the clock divider for the peripheral group 0. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.5.8 PSS Profile PLL Control Register (SYSC_PSSPLLCNTR)

This register controls the Main PLL configuration in the PSS. For PLL stabilization time settings refer to 2.3.5.3 Source clocks.

PSS Profile PLL Control Register (SYSC_PSSPLLCNTR)

Figure 2-25. PSS Profile PLL Control Register (SYSC_PSSPLLCNTR)

SYSC_PSSPLLCNTR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	PLLDIVN[6]	PLLDIVN[5]	PLLDIVN[4]	PLLDIVN[3]	PLLDIVN[2]	PLLDIVN[1]	PLLDIVN[0]	read0	read0	read0	read0	PLLDIVM[3]	PLLDIVM[2]	PLLDIVM[1]	PLLDIVM[0]	read0	read0	read0	read0	read0	read0	PLLDIVL[1]	PLLDIVL[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-26. PSS Profile PLL Control Register (SYSC_PSSPLLCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	read0	-
[22:16]	PLLDIVN	PLL Multiplication-by-N Value These bits control the multiplication value for Main PLL clock. For the PLL programming refer to Main PLL interface. '0000000': PLL input clock is multiplied by 1 '0000001': PLL input clock is multiplied by 2 ... '1111111': PLL input clock is multiplied by 128 Refer to device specific datasheet for Main PLL output frequency range. Do not program the value which falls outside the range. User should program multiplication factor accordingly.
[15:12]	read0	-
[11:8]	PLLDIVM	PLL Division-by-M Value These bits control the clock divider for Main PLL output clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30 To guarantee 50 percent duty cycle from PLL clock, it is recommended to set the PLL division other than division by 1.

Table 2-26. PSS Profile PLL Control Register (SYSC_PSSPLLCNTR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1:0]	PLLDIVL	PLL Input Division Value These bits control the input clock divider for the Main PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.5.9 PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0)

This register controls the SSCG PLL configuration in the PSS. For SSCG-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0)

Figure 2-26. PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0)

SYSC_PSSSSCGCNTR0																															
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**
read0	read0	read0	SSCGDIVP[4]	SSCGDIVP[3]	SSCGDIVP[2]	SSCGDIVP[1]	SSCGDIVP[0]	read0	read0	SSCGDIVN[5]	SSCGDIVN[4]	SSCGDIVN[3]	SSCGDIVN[2]	SSCGDIVN[1]	SSCGDIVN[0]	read0	read0	read0	read0	SSCGDIVM[3]	SSCGDIVM[2]	SSCGDIVM[1]	SSCGDIVM[0]	read0	read0	read0	read0	read0	read0	SSCGDIVL[1]	SSCGDIVL[0]
R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-27. PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	SSCGDIVP	SSCG PLL Multiplication-by-P Value These bits along with SYSC_PSSSSCGCNTR0:SSCGDIVN bits define the multiplication value for SSCG PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_PSSSSCGCN-TR0:SSCGDIVN x 1) '00010':PLL input clock multiplied by (SYSC_PSSSSCGCN-TR0:SSCGDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_PSSSSCGCN-TR0:SSCGDIVN x 31) Refer to device specific datasheet for SSCG PLL output frequency range. User should program multiplication factor accordingly.
[23:22]	read0	-

Table 2-27. PSS Profile SSCG Control Register 0 (SYSC_PSSSSCGCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	SSCGDIVN	SSCG PLL Multiplication-by-N Value These bits along with SYSC_PSSSSCGCNTR0:SSCGDIVP bits define the multiplication value for SSCG PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_PSSSSCGCNTR0:SSCGDIVP x 2) '000011': PLL input clock multiplied by (SYSC_PSSSSCGCNTR0:SSCGDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_PSSSSCGCNTR0:SSCGDIVP x 63) Refer to device specific datasheet for SSCG PLL output frequency range. User should program multiplication factor accordingly.
[15:12]	read0	-
[11:8]	SSCGDIVM	SSCG PLL Division-by-M Value These bits control the clock divider for the division value for SSCG PLL. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30 To guarantee 50 percent duty cycle from SSCG PLL clock, it is recommended to set the SSCG PLL division other than division by 1.
[7:2]	read0	-
[1:0]	SSCGDIVL	SSCG PLL Input Division Value These bits control the input clock divider for the SSCG PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.5.10 PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1)

This register controls the SSCG PLL configuration in the PSS.

PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1)

Figure 2-27. PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1)

SYSC_PSSSSCGCNTR1																															
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**
read0	read0	read0	read0	read0	read0	read0	SSCGSSEN	read0	read0	read0	read0	read0	SSCGFREQ[1]	SSCGFREQ[0]	SSCGMODE	read0	read0	read0	read0	read0	read0	SSCGRATE[9]	SSCGRATE[8]	SSCGRATE[7]	SSCGRATE[6]	SSCGRATE[5]	SSCGRATE[4]	SSCGRATE[3]	SSCGRATE[2]	SSCGRATE[1]	SSCGRATE[0]
R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-28. PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	SSCGSSEN	SSCG PLL Modulation Enable Control These bits enable/disable the modulation for SSCG PLL. '0': SSCG PLL modulation is disabled '1': SSCG PLL modulation is enabled
[23:19]	read0	-
[18:17]	SSCGFREQ	SSCG PLL Modulation Frequency Select These bits select the modulation frequency rate for SSCG PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	SSCGMODE	SSCG PLL Modulator Mode This bit controls the modulation mode of SSCG PLL. '0': SSCG PLL works in down spread mode '1': SSCG PLL works in center spread mode
[15:10]	read0	-

Table 2-28. PSS Profile SSCG Control Register 1 (SYSC_PSSSSCGCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	SSCGRATE	SSCG PLL Modulation Rate Control Value These bits control the modulation rate for SSCG PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.5.11 PSS Profile Graphics PLL Control Register 0 (SYSC_PSSGFXCNTR0)

This register controls the Graphics PLL configuration in the PSS. For GFX-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

PSS Profile Graphics PLL Control Register 0 (SYSC_PSSGFXCNTR0)

Figure 2-28. PSS Profile GFX PLL Control Register 0 (SYSC_PSSGFXCNTR0)

SYSC_PSSGFXCNTR0																															
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**
read0	read0	read0	GFXDIVP[4]	GFXDIVP[3]	GFXDIVP[2]	GFXDIVP[1]	GFXDIVP[0]	read0	read0	GFXDIVN[5]	GFXDIVN[4]	GFXDIVN[3]	GFXDIVN[2]	GFXDIVN[1]	GFXDIVN[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXDIVL[1]	GFXDIVL[0]
R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-29. PSS Profile GFX PLL Control Register 0 (SYSC_PSSGFXCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	GFXDIVP	Graphics PLL Division-by-P Value These bits along with SYSC_PSSGFXCNTR0:GFXDIVN bits define the multiplication value for GFX PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_PSSGFXCN-TR0:GFXDIVN x 1) '00010':PLL input clock multiplied by (SYSC_PSSGFXCN-TR0:GFXDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_PSSGFXCN-TR0:GFXDIVN x 31) Refer to device specific datasheet for GFX PLL output frequency range. User should program multiplication factor accordingly.
[23:22]	read0	-

Table 2-29. PSS Profile GFX PLL Control Register 0 (SYSC_PSSGFXCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	GFXDIVN	Graphics PLL Division-by-N Value These bits along with SYSC_PSSGFXCNTR0:GFXDIVP bits define the multiplication value for GFX PLL clock. For PLL programming refer to SSCG and GFX PLL interface. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_PSSGFXCN-TR0:GFXDIVP x 2) '000011': PLL input clock multiplied by (SYSC_PSSGFXCN-TR0:GFXDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_PSSGFXCN-TR0:GFXDIVP x 63) Refer to device specific datasheet for GFX PLL output frequency range. User should program multiplication factor accordingly.
[15:8]	read0	-
[7:2]	read0	-
[1:0]	GFXDIVL	Graphics PLL Input Division Value These bits control the input clock divider for the GFX PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.5.12 PSS Profile Graphics PLL Control Register 1 (SYSC_PSSGFXCNTR1)

This register controls the Graphics PLL configuration in the PSS.

PSS Profile Graphics PLL Control Register 1 (SYSC_PSSGFXCNTR1)

Figure 2-29. PSS Profile GFX PLL Control Register 1 (SYSC_PSSGFXCNTR1)

SYSC_PSSGFXCNTR1																															
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**
read0	read0	read0	read0	read0	read0	read0	GFXSSEN	read0	read0	read0	read0	read0	GFXFREQ[1]	GFXFREQ[0]	GFXMODE	read0	read0	read0	read0	read0	read0	GFXRATE[9]	GFXRATE[8]	GFXRATE[7]	GFXRATE[6]	GFXRATE[5]	GFXRATE[4]	GFXRATE[3]	GFXRATE[2]	GFXRATE[1]	GFXRATE[0]
R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-30. PSS Profile GFX PLL Control Register 1 (SYSC_PSSGFXCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	GFXSSEN	GFX PLL Modulation Enable Control These bits enable/disable the modulation for GFX PLL. '0': GFX PLL modulation is disabled '1': GFX PLL modulation is enabled
[23:19]	read0	-
[18:17]	GFXFREQ	Graphics PLL Modulation Frequency Select These bits select the modulation frequency rate for GFX PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	GFXMODE	Graphics PLL Modulator Mode This bit controls the modulation mode of GFX PLL. '0': GFX PLL works in down spread mode '1': GFX PLL works in center spread mode
[15:10]	read0	-

Table 2-30. PSS Profile GFX PLL Control Register 1 (SYSC_PSSGFXCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	GFXRATE	Graphics PLL Modulation Rate Control Value These bits control the modulation rate for GFX PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.5.13 PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR)

This register controls the LVD setting in the PSS.

PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR)

Figure 2-30. PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR)

SYSC_PSSLVDCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SV12[2]	SV12[1]	SV12[0]	LVDR12	LVDE12	read0	read0	read0	SV33[2]	SV33[1]	SV33[0]	LVDR33	LVDE33	read0	read0	read0	SV50[2]	SV50[1]	SV50[0]	LVDR50	LVDE50
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1

Table 2-31. PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:21]	read0	-
[20:18]	SV12	Reference Voltage Selection for 1.2 V LVD These bits control analog threshold for Low Voltage Detection on 1.2 V core supply. '000': 0.5 V '001': 0.6 V '011': 0.7 V '010': 0.8 V '110': 0.9 V '111': 1.0 V '101': 1.1 V '100': 1.2 V On reset, 0.8 V is chosen as reference voltage. This bit can set SYSC_SYSSTSR:IPPAPSS.
[17]	LVDR12	Low Voltage Detector Reset/Interrupt Select for 1.2 V This bit is used to enable interrupt/reset for LVD on 1.2 V core supply. '0': Low Voltage Detect on 1.2 V sets interrupt flag SYSC_SYSERRR:LVD12IF '1': Low Voltage Detect on 1.2 V generates reset **Note:** The generation of a low voltage interrupt requires, that the CLK_CFG_PD1 is enabled. Thus SYSC_PSSCKER:ENCFGPD1 must be set to '1'.

Table 2-31. PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[16]	LVDE12	Low Voltage Detector Enable for 1.2 V This bit is used to enable/disable LVD functionality on 1.2 V core supply. '0': LVD on 1.2 V disabled '1': LVD on 1.2 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over. This bit can set SYSC_SYSSTSR:IPPAPSS.
[15:13]	read0	-
[12:10]	SV33	Reference Voltage Selection for 3.3 V LVD These bits control analog threshold for Low Voltage Detection on 3.3 V Graphics IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V Others: Reserved User should not configure reserved values to avoid unexpected reset. On reset, 2.6 V is chosen as reference voltage. This bit can set SYSC_SYSSTSR:IPPAPSS.
[9]	LVDR33	Low Voltage Detector Reset/Interrupt Select for 3.3 V This bit is used to enable interrupt/reset for LVD on 3.3 V Graphics IO supply. '0': Low Voltage Detect on 3.3 V sets interrupt flag SYSC_SYSERRR:LVD33IF '1': Low Voltage Detect on 3.3 V generates reset **Note:** The generation of a low voltage interrupt requires, that the CLK_CFG_PD1 is enabled. Thus SYSC_PSSCKER:ENCFGPD1 must be set to '1'.
[8]	LVDE33	Low Voltage Detector Enable for 3.3 V This bit is used to enable/disable LVD functionality on 3.3 V Graphics IO supply. '0': LVD on 3.3 V disabled '1': LVD on 3.3 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over. This bit can set SYSC_SYSSTSR:IPPAPSS.
[7:5]	read0	-

Table 2-31. PSS Profile Low Voltage Detector Configuration Register (SYSC_PSSLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[4:2]	SV50	Reference Voltage Selection for 5.0 V LVD These bits control analog threshold for Low Voltage Detection on 5.0 V IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V '110': 3.7 V '111': 3.9 V '101': 4.1 V '100': 4.3 V On reset, 2.6 V is chosen as reference voltage. This bit can set SYSC_SYSSTSR:IPPAPSS.
[1]	LVDR50	Low Voltage Detector Reset/Interrupt Select for 5.0 V This bit is used to enable interrupt/reset for LVD on 5.0 V IO supply. '0': Low Voltage Detect on 5.0 V sets interrupt flag SYSC_SYSERRR:LVD50IF '1': Low Voltage Detect on 5.0 V generates reset **Note:** The generation of a low voltage interrupt requires, that the CLK_CFG_PD1 is enabled. Thus SYSC_PSSCKER:ENCFGPD1 must be set to '1'.
[0]	LVDE50	Low Voltage Detector Enable for 5.0 V This bit is used to enable/disable LVD functionality on 5.0 V IO supply. '0': LVD on 5.0 V disabled '1': LVD on 5.0 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over. This bit can set SYSC_SYSSTSR:IPPAPSS.

2.2.5.14 PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR)

This register controls the CSV settings in the PSS.

PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR)

Figure 2-31. PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR)

SYSC_PSSCSVCFGR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GPCSVE	SPCSVE	MPCSVE	SOCSVE	MOCSVE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-32. PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:5]	read0	-
[4]	GPCSVE	Graphics PLL Clock Supervisor Enable This bit enables clock supervision on Graphics PLL clock. '0': Graphics PLL clock supervision disabled '1': Graphics PLL clock supervision enabled Disabling Graphics PLL clock supervision when the Graphics PLL is still enabled is invalid condition. Device state logic rejects these settings and SYSC_SYSSTSR:IPPAPSS error flag is set.
[3]	SPCSVE	SSCG PLL Clock Supervisor Enable This bit enables clock supervision on SSCG PLL clock. '0': SSCG PLL clock supervision disabled '1': SSCG PLL clock supervision enabled Disabling SSCG PLL clock supervision when the SSCG PLL is still enabled is invalid condition. Device state logic rejects these settings and SYSC_SYSSTSR:IPPAPSS error flag is set.
[2]	MPCSVE	Main PLL Clock Supervisor Enable This bit enables clock supervision on Main PLL clock. '0': Main PLL clock supervision disabled '1': Main PLL clock supervision enabled Disabling Main PLL clock supervision when the Main PLL is still enabled is invalid condition. Device state logic rejects these settings and SYSC_SYSSTSR:IPPAPSS error flag is set.

Table 2-32. PSS Profile Clock Supervisor Configuration Register (SYSC_PSSCSVCFGR) bits

Bit Position	Bit Field Name	Bit Description
[1]	SOCSVE	Sub Oscillator Clock Supervisor Enable This bit enables clock supervision on Sub clock. '0': Sub clock supervision disabled '1': Sub clock supervision enabled Disabling Sub clock supervision when the Sub clock is still enabled is invalid condition. Device state logic rejects these settings and SYSC_SYSSTSR:IPPAPSS error flag is set.
[0]	MOCSVE	Main Oscillator Clock Supervisor Enable This bit enables clock supervision on Main clock. '0': Main clock supervision disabled '1': Main clock supervision enabled Disabling Main clock supervision when the Main clock is still enabled is invalid condition. Device state logic rejects these settings and SYSC_SYSSTSR:IPPAPSS error flag is set.

2.2.5.15 PSS Enable Register (SYSC_PSSENR)

This register is used to enable PSS profile settings if CPU goes into standby mode.

PSS Enable Register (SYSC_PSSENR)

Figure 2-32. PSS Enable Register (SYSC_PSSENR)

SYSC_PSSENR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	PSSEN[7]	PSSEN[6]	PSSEN[5]	PSSEN[4]	PSSEN[3]	PSSEN[2]	PSSEN[1]	PSSEN[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-33. PSS Enable Register (SYSC_PSSENR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:0]	PSSEN	PSS Switch Enable Register bits These bits when configured as 0xBA enables state control logic to switch device state to PSS. These bits need to be configured after PSS and RUN profile configuration. Profile validation checks are enabled when this bit is set to '1'. On profile check failure, these bits are reset to '0' by hardware. Updating of RUN and PSS profile register by the user after configuration of these bits will reset the PSSEN bits by the hardware. These bits can be cleared by the user at any time. If enabled and CPU goes into standby state due to WFI instruction execution the PSS switching starts. Writing value other than 0xBA will not update the register and sets SYSC_SYSERRR:PSSENEIF flag.

2.2.6 APPLIED profile register set

APPLIED profile register set shows the current applied settings to the device. As application of profile takes time, the actual device state may be different than the APPLIED profile register sets. The actual state of the device is reflected in the profile status register group. The applied profile can be used by the user to know the previous profile settings applied to device. This register set gets initialized on hard resets.

2.2.6.1 APPLIED Profile Power Domain Configuration Register (SYSC_APPPDCFGR)

This register shows the current applied power domain settings.

APPLIED Profile Power Domain Configuration Register (SYSC_APPPDCFGR)

Figure 2-33. APPLIED Profile Power Domain Configuration Register (SYSC_APPPDCFGR)

SYSC_APPPDCFGR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PD5ON	PD4ON	PD3ON	PD2ON
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 2-34. APPLIED Profile Power Domain Configuration Register (SYSC_APPPDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	PD5ON	Power Domain 5 Configuration bit This bit shows the current power domain 5 settings applied to the device. '0': Power domain 5 switch OFF '1': Power domain 5 switch ON
[2]	PD4ON	Power Domain 4 Configuration bit This bit shows the current power domain 4 settings applied to the device. '0': Power domain 4 switch OFF '1': Power domain 4 switch ON
[1]	PD3ON	Power Domain 3 Configuration bit This bit shows the current power domain 3 settings applied to the device. '0': Power domain 3 switch OFF '1': Power domain 3 switch ON
[0]	PD2ON	Power Domain 2 Configuration bit This bit shows the current power domain 2 settings applied to the device. '0': Power domain 2 switch OFF '1': Power domain 2 switch ON

2.2.6.2 APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER)

This register shows the current applied source clock settings.

APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER)

Figure 2-34. APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER)

SYSC_APPCKSRER															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXPLLEN	SSCGPLLEN	MAINPLLEN	SOSCEN	MOSCEN	RCOSCEN	SRCOSCEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Table 2-35. APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7]	read0	-
[6]	GFXPLLEN	GFX PLL Enable Control bit This bit shows the current GFX PLL oscillation circuit settings applied to the device. '0': GFX PLL oscillator disable '1': GFX PLL oscillator enable
[5]	SSCGPLLEN	SSCG PLL Enable Control bit This bit shows the current SSCG PLL oscillation circuit settings applied to the device. '0': SSCG PLL oscillator disable '1': SSCG PLL oscillator enable
[4]	MAINPLLEN	Main PLL Enable Control bit This bit shows the current Main PLL oscillation circuit settings applied to the device. '0': Main PLL oscillator disable '1': Main PLL oscillator enable
[3]	SOSCEN	Sub Oscillator Enable Control bit This bit shows the current Sub oscillation circuit settings applied to device. '0': Sub oscillator disable '1': Sub oscillator enable

Table 2-35. APPLIED Profile Clock Source Enable Register (SYSC_APPCKSRER) bits

Bit Position	Bit Field Name	Bit Description
[2]	MOSCEN	Main Oscillator Enable Control bit This bit shows the current Main oscillation circuit settings applied to device. '0': Main oscillator disable '1': Main oscillator enable
[1]	RCOSCEN	RC Oscillator Enable Control bit This bit shows the current RC oscillation circuit settings applied to the device. '0': RC oscillator disable '1': RC oscillator enable
[0]	SRCOSCEN	Slow RC Oscillator Enable Control bit This bit shows the current Slow RC oscillation circuit settings applied to the device. '0': Slow RC oscillator disable '1': Slow RC oscillator enable

2.2.6.3 APPLIED Profile Clock Select Register (SYSC_APPCKSELR)

This register shows the current applied clock selection multiplexer settings.

APPLIED Profile Clock Select Register (SYSC_APPCKSELR)

Figure 2-35. APPLIED Profile Clock Select Register (SYSC_APPCKSELR)

SYSC_APPCKSELR																															
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**
read0	read0	SPICSL[1]	SPICSL[0]	read0	read0	PIXCSL[1]	PIXCSL[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI3CSL[2]	PERI3CSL[1]	PERI3CSL[0]	read0	PERI1CSL[2]	PERI1CSL[1]	PERI1CSL[0]	read0	PERI0CSL[2]	PERI0CSL[1]	PERI0CSL[0]	read0	SYSCSL[2]	SYSCSL[1]	SYSCSL[0]
R0	R0	R	R	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R0	R	R	R	R0	R	R	R	R0	R	R	R
0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-36. APPLIED Profile Clock Select Register (SYSC_APPCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[31:30]	read0	-
[29:28]	SPICSL	SPI Clock Select for Graphics Subsystem These bits show the applied clock for the Graphics Subsystem SPI clock. '00': SPI clock (Main PLL clock) is selected '01': External clock is selected 10 to 11: Clock is tied low
[27:26]	read0	-
[25:24]	PIXCSL	Pixel Clock Select for Graphics Subsystem These bits show the applied clock for the Graphics Subsystem pixel clock. '00': GFX bus clock is selected '01': External clock is selected '10': GFX PLL clock is selected '11': Clock is tied low
[23:16]	read0	-
[15]	read0	-

Table 2-36. APPLIED Profile Clock Select Register (SYSC_APPCKSELR) bits

Bit Position	Bit Field Name	Bit Description
[14:12]	PERI3CSL	Peripheral Group 3 Clock Select Control bit These bits show the applied clock for the peripheral group 3. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low
[11]	read0	-
[10:8]	PERI1CSL	Peripheral Group 1 Clock Select Control bit These bits show the applied clock for the peripheral group 1. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low
[7]	read0	-
[6:4]	PERI0CSL	Peripheral Group 0 Clock Select Control bit These bits show the applied clock for the peripheral group 0. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low
[3]	read0	-
[2:0]	SYSCSL	System Clock Select Control bit These bits show the applied clock for the core group. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low

2.2.6.4 APPLIED Profile Clock Enable Register (SYSC_APPCKER)

This register shows the current applied clock gate settings.

APPLIED Profile Clock Enable Register (SYSC_APPCKER)

Figure 2-36. APPLIED Profile Clock Enable Register (SYSC_APPCKER)

SYSC_APPCKER																															
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**
read0	ENSPIPD5	ENPIXPD5	ENGFXPD5	read0	read0	read0	ENCFGPD4	read0	read0	ENEXTBUSPD3	ENSPIPD3	ENMEMEPD3	ENHPMPD3	ENTRACEPD3	ENSYSPD3	read0	read0	read0	read0	ENPERI4PD2	ENPERI3PD2	read0	ENPERI1PD2	ENPERI0PD2	ENDMAPD2	ENTRACEPD2	ENHPMPD2	read0	read0	read0	ENCFGPD1
R0	R	R	R	R0	R0	R0	R	R0	R0	R	R	R	R	R	R	R0	R0	R0	R0	R	R	R0	R	R	R	R	R	R0	R0	R0	R
0	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	1

Table 2-37. APPLIED Profile Clock Enable Register (SYSC_APPCKER) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30]	ENSPIPD5	Enable Graphics SPI Clock for Power Domain 5 This bit shows the applied SPI clock to the Graphics Subsystem. '0': SPI clock to the Graphics Subsystem is disabled '1': SPI clock to the Graphics Subsystem is enabled
[29]	ENPIXPD5	Enable Graphics PLL Clock for Power Domain 5 This bit shows the applied pixel clock to the Graphics Subsystem. '0': Pixel clock to the Graphics Subsystem is disabled '1': Pixel clock to the Graphics Subsystem is enabled
[28]	ENGFXPD5	Enable Graphics Bus Clock for Power Domain 5 This bit is shows the applied bus clock to the Graphics Subsystem. '0': Bus clock to the Graphics Subsystem is disabled '1': Bus clock to the Graphics Subsystem is enabled
[27:25]	read0	-
[24]	ENCFGPD4	Enable Configuration Clock for Power Domain 4 This bit shows the applied clocks to the Retention RAM. '0': Retention RAM clock is disabled '1': Retention RAM clock is enabled
[23:22]	read0	-

Table 2-37. APPLIED Profile Clock Enable Register (SYSC_APPCKER) bits

Bit Position	Bit Field Name	Bit Description
[21]	ENEXTBUSPD3	Enable External Bus Clock for Power Domain 3 This bit shows the applied clock status to the external bus peripheral. '0': Clock to external bus peripheral disabled '1': Clock to external bus peripheral enabled
[20]	ENSPIPD3	Enable SPI Clock of HSSPI for Power Domain 3 This bit shows the applied SPI clock to HSSPI module. '0': SPI clock of HSSPI is disabled '1': SPI clock of HSSPI is enabled
[19]	ENMEMEPD3	Enable Memory External Clock for Power Domain 3 This bit shows the applied clock status to the external memory group. '0': Clock to external memory group disabled '1': Clock to external memory group enabled
[18]	ENHPMPD3	Enable HPM Clock for Power Domain 3 This bit shows the applied clock status to the AXI port of TCFLASH interface. '0': Clock to the AXI port of TCFLASH interface disabled '1': Clock to the AXI port of TCFLASH interface enabled
[17]	ENTRACEPD3	Enable Trace Clock for Power Domain 3 This bit shows the applied clock status to the debug and trace group in power domain 3. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled
[16]	ENSYSPD3	Enable System Clock for Power Domain 3 This bit shows the applied clock status to the core group. '0': Clock to core group disabled '1': Clock to core group enabled
[15:12]	read0	-
[11]	ENPERI4PD2	Enable Peripheral Group 4 Clock for Power Domain 2 This bit shows the applied clock status to the peripheral group 4. '0': Clock to peripheral group 4 disabled '1': Clock to peripheral group 4 enabled
[10]	ENPERI3PD2	Enable Peripheral Group 3 Clock for Power Domain 2 This bit shows the applied clock status to the peripheral group 3. '0': Clock to peripheral group 3 disabled '1': Clock to peripheral group 3 enabled
[9]	read0	-
[8]	ENPERI1PD2	Enable Peripheral Group 1 Clock for Power Domain 2 This bit shows the applied clock status to the peripheral group 1. '0': Clock to peripheral group 1 disabled '1': Clock to peripheral group 1 enabled

Table 2-37. APPLIED Profile Clock Enable Register (SYSC_APPCKER) bits

Bit Position	Bit Field Name	Bit Description
[7]	ENPERI0PD2	Enable Peripheral Group 0 Clock for Power Domain 2 This bit shows the applied clock status to the peripheral group 0. '0': Clock to peripheral group 0 disabled '1': Clock to peripheral group 0 enabled
[6]	ENDMAPD2	Enable DMA Clock for Power Domain 2 This bit shows the applied clock status to the DMA in power domain 2. '0': Clock to DMA disabled '1': Clock to DMA enabled
[5]	ENTRACEPD2	Enable Trace Clock for Power Domain 2 This bit shows the applied clock status to the debug and trace group in power domain 2. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled
[4]	ENHPMPD2	Enable HPM Clock for Power Domain 2 This bit shows the applied clock status to the HPM bus matrix. '0': Clock to HPM bus matrix disabled '1': Clock to HPM bus matrix enabled
[3:1]	read0	-
[0]	ENCFGPD1	Enable Configuration Clock for Power Domain 1 This bit shows the applied clock status to the configuration group. '0': Clock to configuration group disabled '1': Clock to configuration group enabled

2.2.6.5 APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0)

This register shows the current applied clock division ratio settings.

APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0)

Figure 2-37. APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0)

SYSC_APPCKDIVR0																														
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**
read0	read0	read0	read0	read0	read0	CFGDIV[1]	CFGDIV[0]	read0	read0	read0	read0	read0	read0	HPMDIV[1]	HPMDIV[0]	read0	read0	read0	read0	read0	read0	TRACEDIV[1]	TRACEDIV[0]	read0	read0	read0	read0	SYSDIV[3]	SYSDIV[2]	SYSDIV[1]	SYSDIV[0]
R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-38. APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	CFGDIV	Configuration Clock Division Value These bits show the applied clock divider for the configuration group clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	HPMDIV	HPM Clock Division Value These bits show the applied clock divider for the HPM clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-
[9:8]	TRACEDIV	Trace Clock Division Value These bits show the applied clock divider for the debug and trace clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8

Table 2-38. APPLIED Profile Clock Divider Register 0 (SYSC_APPCKDIVR0) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3:0]	SYSDIV	System Clock Division Value These bits show the applied clock divider for the system clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.6.6 APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1)7

This register shows the current applied clock division ratio settings.

APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1)

Figure 2-38. APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1)

SYSC_APPCKDIVR1																													
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**
read0	read0	read0	read0	read0	read0	GFXDIV[1]	GFXDIV[0]	read0	read0	read0	read0	read0	read0	PERI4DIV[1]	PERI4DIV[0]	read0	read0	read0	read0	read0	read0	MEMEDIV[1]	MEMEDIV[0]	read0	read0	read0	read0	read0	EXTBUSDIV[2]	EXTBUSDIV[1]	EXTBUSDIV[0]
R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-39. APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	GFXDIV	Graphics Clock Division Value These bits show the applied clock divider for the Graphics Subsystem bus clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	PERI4DIV	Peripheral 4 Clock Division Value These bits show the applied clock divider for the peripheral group 4 clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-

Table 2-39. APPLIED Profile Clock Divider Register 1 (SYSC_APPCKDIVR1) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	MEMEDIV	Memory External Clock Division Value These bits show the applied clock divider for the external memory group (HSSPI module) clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[7:3]	read0	-
[2:0]	EXTBUSDIV	External Bus Clock Division Value These bits show the applied clock divider for the external bus clock. For more details refer to Figure 2-132. '000': Clock division is bypassed (division by 1) '001': Divided by 2 '010': Divided by 4 '011': Divided by 8 ... '110': Divided by 64, '111': Divided by 128,

2.2.6.7 APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2)

This register shows the current applied clock division ratio settings.

APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2)

Figure 2-39. APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2)

SYSC_APPCKDIVR2																															
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**
read0	read0	read0	read0	PERI3DIV[3]	PERI3DIV[2]	PERI3DIV[1]	PERI3DIV[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI1DIV[3]	PERI1DIV[2]	PERI1DIV[1]	PERI1DIV[0]	read0	read0	read0	read0	PERI0DIV[3]	PERI0DIV[2]	PERI0DIV[1]	PERI0DIV[0]
R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-40. APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:24]	PERI3DIV	Peripheral Group 3 Clock Division Value These bits show the applied clock divider for the peripheral group 3. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[23:16]	read0	-
[15:12]	read0	-
[11:8]	PERI1DIV	Peripheral Group 1 Clock Division Value These bits show the applied clock divider for the peripheral group 1. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:4]	read0	-

Table 2-40. APPLIED Profile Clock Divider Register 2 (SYSC_APPCKDIVR2) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	PERI0DIV	Peripheral Group 0 Clock Division Value These bits show the applied clock divider for the peripheral group 0. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.6.8 APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR)

This register shows the current applied Main PLL configuration settings. For PLL stabilization time settings refer to 2.3.5.3 Source clocks.

APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR)

Figure 2-40. APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR)

SYSC_APPPLLCNTR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	PLLDIVN[6]	PLLDIVN[5]	PLLDIVN[4]	PLLDIVN[3]	PLLDIVN[2]	PLLDIVN[1]	PLLDIVN[0]	read0	read0	read0	read0	PLLDIVM[3]	PLLDIVM[2]	PLLDIVM[1]	PLLDIVM[0]	read0	read0	read0	read0	read0	read0	PLLDIVL[1]	PLLDIVL[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-41. APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	read0	-
[22:16]	PLLDIVN	PLL Multiplication-by-N Value These bits show the applied multiplication value for Main PLL clock. '0000000': PLL input clock is multiplied by 1 '0000001': PLL input clock is multiplied by 2 ... '1111111': PLL input clock is multiplied by 128 Refer to device specific datasheet for Main PLL output frequency range.
[15:12]	read0	-
[11:8]	PLLDIVM	PLL Division-by-M Value These bits show the applied clock divider for Main PLL output clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:2]	read0	-

Table 2-41. APPLIED Profile PLL Control Register (SYSC_APPPLLCNTR) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	PLLDIVL	PLL Input Division Value These bits show the applied clock divider for the Main PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.6.9 APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0)

This register shows the current applied SSCG PLL configuration settings. For SSCG-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0)

Figure 2-41. APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0)

SYSC_APPSSCGCNTR0																															
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**
read0	read0	read0	SSCGDIVP[4]	SSCGDIVP[3]	SSCGDIVP[2]	SSCGDIVP[1]	SSCGDIVP[0]	read0	read0	SSCGDIVN[5]	SSCGDIVN[4]	SSCGDIVN[3]	SSCGDIVN[2]	SSCGDIVN[1]	SSCGDIVN[0]	read0	read0	read0	read0	SSCGDIVM[3]	SSCGDIVM[2]	SSCGDIVM[1]	SSCGDIVM[0]	read0	read0	read0	read0	read0	read0	SSCGDIVL[1]	SSCGDIVL[0]
R0	R0	R0	R	R	R	R	R	R0	R0	R	R	R	R	R	R	R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-42. APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	SSCGDIVP	SSCG PLL Multiplication-by-P Value These bits along with SYSC_APPSSCGCNTR0:SSCGDIVN bits show the applied multiplication value for SSCG PLL clock. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_APPSSCGCN-TR0:SSCGDIVN x 1) '00010':PLL input clock multiplied by (SYSC_APPSSCGCN-TR0:SSCGDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_APPSSCGCN-TR0:SSCGDIVN x 31)
[23:22]	read0	-
[21:16]	SSCGDIVN	SSCG PLL Multiplication-by-N Value These bits along with SYSC_APPSSCGCNTR0:SSCGDIVP bits show the applied multiplication value for SSCG PLL clock. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_APPSSCGCNTR0:SSCGDIVP x 2) '000011': PLL input clock multiplied by (SYSC_APPSSCGCN-TR0:SSCGDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_APPSSCGCN-TR0:SSCGDIVP x 63)
[15:12]	read0	-

Table 2-42. APPLIED Profile SSCG Control Register 0 (SYSC_APPSSCGCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[11:8]	SSCGDIVM	SSCG PLL Division-by-M Value These bits show the applied clock divider for the division value for SSCG PLL. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:2]	read0	-
[1:0]	SSCGDIVL	SSCG PLL Input Division Value These bits show the applied clock divider for the SSCG PLL Macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.6.10 APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1)

This register shows the current applied SSCG PLL configuration settings.

APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1)

Figure 2-42. APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1)

SYSC_APPSSCGCNTR1																															
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**
read0	read0	read0	read0	read0	read0	read0	SSCGSSEN	read0	read0	read0	read0	read0	SSCGFREQ[1]	SSCGFREQ[0]	SSCGMODE	read0	read0	read0	read0	read0	read0	SSCGRATE[9]	SSCGRATE[8]	SSCGRATE[7]	SSCGRATE[6]	SSCGRATE[5]	SSCGRATE[4]	SSCGRATE[3]	SSCGRATE[2]	SSCGRATE[1]	SSCGRATE[0]
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-43. APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	SSCGSSEN	SSCG PLL Modulation Enable Control These bits show the applied modulation setting for SSCG PLL. '0': SSCG PLL modulation is disabled '1': SSCG PLL modulation is enabled
[23:19]	read0	-
[18:17]	SSCGFREQ	SSCG PLL Modulation Frequency Select These bits show the applied modulation frequency rate for SSCG PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	SSCGMODE	SSCG PLL Modulator Mode This bit shows the applied modulation mode of SSCG PLL. '0': SSCG PLL works in down spread mode '1': SSCG PLL works in center spread mode
[15:10]	read0	-

Table 2-43. APPLIED Profile SSCG Control Register 1 (SYSC_APPSSCGCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	SSCGRATE	SSCG PLL Modulation Rate Control Value These bits show the applied modulation rate for SSCG PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved. Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.6.11 APPLIED Profile Graphics PLL Control Register 0 (SYSC_APPGFXCNTR0)

This register shows the current applied Graphics PLL configuration settings. For GFX-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

APPLIED Profile Graphics PLL Control Register 0 (SYSC_APPGFXCNTR0)

Figure 2-43. APPLIED Profile GFX PLL Control Register 0 (SYSC_APPGFXCNTR0)

SYSC_APPGFXCNTR0																															
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**
read0	read0	read0	GFXDIVP[4]	GFXDIVP[3]	GFXDIVP[2]	GFXDIVP[1]	GFXDIVP[0]	read0	read0	GFXDIVN[5]	GFXDIVN[4]	GFXDIVN[3]	GFXDIVN[2]	GFXDIVN[1]	GFXDIVN[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXDIVL[1]	GFXDIVL[0]
R0	R0	R0	R	R	R	R	R	R0	R0	R	R	R	R	R	R	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-44. APPLIED Profile GFX PLL Control Register 0 (SYSC_APPGFXCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	GFXDIVP	Graphics PLL Division-by-P Value These bits along with SYSC_APPGFXCNTR0:GFXDIVN bits show the applied multiplication value for GFX PLL clock. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_APPGFXCN-TR0:GFXDIVN x 1) '00010':PLL input clock multiplied by (SYSC_APPGFXCN-TR0:GFXDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_APPGFXCN-TR0:GFXDIVN x 31)
[23:22]	read0	-
[21:16]	GFXDIVN	Graphics PLL Division-by-N Value These bits along with SYSC_APPGFXCNTR0:GFXDIVP bits show the applied multiplication value for GFX PLL clock. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_APPGFXCN-TR0:GFXDIVP x 2) '000011': PLL input clock multiplied by (SYSC_APPGFXCN-TR0:GFXDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_APPGFXCN-TR0:GFXDIVP x 63)
[15:8]	read0	-

Table 2-44. APPLIED Profile GFX PLL Control Register 0 (SYSC_APPGFXCNTR0) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1:0]	GFXDIVL	Graphics PLL Input Division Value These bits show the applied clock divider for the GFX PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.6.12 APPLIED Profile Graphics PLL Control Register 1 (SYSC_APPGFXCNTR1)

This register shows the current applied Graphics PLL configuration settings.

APPLIED Profile Graphics PLL Control Register 1 (SYSC_APPGFXCNTR1)

Figure 2-44. APPLIED Profile GFX PLL Control Register 1 (SYSC_APPGFXCNTR1)

SYSC_APPGFXCNTR1																															
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**
read0	read0	read0	read0	read0	read0	read0	GFXSSEN	read0	read0	read0	read0	read0	GFXFREQ[1]	GFXFREQ[0]	GFXMODE	read0	read0	read0	read0	read0	read0	GFXRATE[9]	GFXRATE[8]	GFXRATE[7]	GFXRATE[6]	GFXRATE[5]	GFXRATE[4]	GFXRATE[3]	GFXRATE[2]	GFXRATE[1]	GFXRATE[0]
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-45. APPLIED Profile GFX PLL Control Register 1 (SYSC_APPGFXCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	GFXSSEN	GFX PLL Modulation Enable Status These bits show the applied modulation setting for GFX PLL. '0': GFX PLL modulation is disabled '1': GFX PLL modulation is enabled
[23:19]	read0	-
[18:17]	GFXFREQ	Graphics PLL Modulation Frequency Select These bits show the applied modulation frequency rate for GFX PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	GFXMODE	Graphics PLL Modulator Mode This bit shows the applied modulation mode of GFX PLL. '0': GFX PLL works in down spread mode '1': GFX PLL works in center spread mode
[15:10]	read0	-

Table 2-45. APPLIED Profile GFX PLL Control Register 1 (SYSC_APPGFXCNTR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	GFXRATE	Graphics PLL Modulation Rate Status Value These bits show the applied modulation rate for GFX PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.6.13 APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR)

This register shows the current applied LVD setting.

APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR)

Figure 2-45. APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR)

SYSC_APPLVDCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SV12[2]	SV12[1]	SV12[0]	LVDR12	LVDE12	read0	read0	read0	SV33[2]	SV33[1]	SV33[0]	LVDR33	LVDE33	read0	read0	read0	SV50[2]	SV50[1]	SV50[0]	LVDR50	LVDE50
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R0	R0	R0	R	R	R	R	R	R0	R0	R0	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1

Table 2-46. APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:21]	read0	-
[20:18]	SV12	Reference Voltage Selection for 1.2 V LVD These bits show the applied analog threshold for Low Voltage Detection on 1.2 V core supply. '000': 0.5 V '001': 0.6 V '011': 0.7 V '010': 0.8 V '110': 0.9 V '111': 1.0 V '101': 1.1 V '100': 1.2 V On reset, 0.8 V is chosen as reference voltage.
[17]	LVDR12	Low Voltage Detector Reset/Interrupt Select for 1.2 V This bit shows the applied settings of LVD reset/interrupt on 1.2 V core supply. '0': Low Voltage Detect on 1.2 V sets interrupt flag SYSC_SYSERRR:LVD12IF '1': Low Voltage Detect on 1.2 V generates reset
[16]	LVDE12	Low Voltage Detector Enable for 1.2 V This bit shows the applied LVD functionality on 1.2 V core supply. '0': LVD on 1.2 V disabled '1': LVD on 1.2 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.

Table 2-46. APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:13]	read0	-
[12:10]	SV33	Reference Voltage Selection for 3.3 V LVD These bits show the applied analog threshold for Low Voltage Detection on 3.3 V Graphics IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V On reset, 2.6 V is chosen as reference voltage.
[9]	LVDR33	Low Voltage Detector Reset/Interrupt Select for 3.3 V This bit shows the applied settings of LVD reset/interrupt on 3.3 V graphics IO supply. '0': Low Voltage Detect on 3.3 V sets interrupt flag SYSC_SYSERRR:LVD33IF '1': Low Voltage Detect on 3.3 V generates reset
[8]	LVDE33	Low Voltage Detector Enable for 3.3 V This bit shows the applied LVD functionality on 3.3 V Graphics IO supply. '0': LVD on 3.3 V disabled '1': LVD on 3.3 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.
[7:5]	read0	-
[4:2]	SV50	Reference Voltage Selection for 5.0 V LVD These bits show the applied analog threshold for Low Voltage Detection on 5.0 V IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V '110': 3.7 V '111': 3.9 V '101': 4.1 V '100': 4.3 V On reset, 2.6 V is chosen as reference voltage.
[1]	LVDR50	Low Voltage Detector Reset/Interrupt Select for 5.0 V This bit shows the applied settings of LVD reset/interrupt on 5.0 V IO supply. '0': Low Voltage Detect on 5.0 V sets interrupt flag SYSC_SYSERRR:LVD50IF '1': Low Voltage Detect on 5.0 V generates reset

Table 2-46. APPLIED Profile Low Voltage Detector Configuration Register (SYSC_APPLVDCFGR) bits

Bit Position	Bit Field Name	Bit Description
[0]	LVDE50	Low Voltage Detector Enable for 5.0 V This bit shows the applied LVD functionality on 5.0 V IO supply. '0': LVD on 5.0 V disabled '1': LVD on 5.0 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.

2.2.6.14 APPLIED Profile Clock Supervisor Configuration Register (SYSC_APPCSVCFGR)

This register shows the current applied CSV settings.

APPLIED Profile Clock Supervisor Configuration Register (SYSC_APPCSVCFGR)

Figure 2-46. APPLIED Profile Clock Supervisor Configuration Register (SYSC_APPCSVCFGR)

SYSC_APPCSVCFGR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GPCSVE	SPCSVE	MPCSVE	SOCSVE	MOCSVE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-47. APPLIED Profile Clock Supervisor Configuration Register (SYSC_APPCSVCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:5]	read0	-
[4]	GPCSVE	Graphics PLL Clock Supervisor Enable This bit shows clock supervision on Graphics PLL clock. '0': Graphics PLL clock supervision disabled '1': Graphics PLL clock supervision enabled
[3]	SPCSVE	SSCG PLL Clock Supervisor Enable This bit shows clock supervision on SSCG PLL clock. '0': SSCG PLL clock supervision disabled '1': SSCG PLL clock supervision enabled
[2]	MPCSVE	Main PLL Clock Supervisor Enable This bit shows clock supervision on Main PLL clock. '0': Main PLL clock supervision disabled '1': Main PLL clock supervision enabled
[1]	SOCSVE	Sub Oscillator Clock Supervisor Enable This bit shows clock supervision on Sub clock. '0': Sub clock supervision disabled '1': Sub clock supervision enabled
[0]	MOCSVE	Main Oscillator Clock Supervisor Enable This bit shows clock supervision on Main clock. '0': Main clock supervision disabled '1': Main clock supervision enabled

2.2.7 Profile status register set

Profile status register set shows the actual device status. User can read this register set to know the actual device state at that time. This register sets are dynamically updated by the hardware as and when the device state changes in response to new profile application (RUN or PSS). This register set gets initialized on hard resets.

2.2.7.1 Power Domain Status Register (SYSC_PDSTSR)

This register shows the power domain status.

Power Domain Status Register (SYSC_PDSTSR)

Figure 2-47. Power Domain Status Register (SYSC_PDSTSR)

SYSC_PDSTSR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PD5ON	PD4ON	PD3ON	PD2ON
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 2-48. Power Domain Status Register (SYSC_PDSTSR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	PD5ON	Power Domain 5 Status bit This bit shows current power domain 5 status. '0': Power domain 5 switched OFF '1': Power domain 5 switched ON
[2]	PD4ON	Power Domain 4 Status bit This bit shows current power domain 4 status. '0': Power domain 4 switched OFF '1': Power domain 4 switched ON
[1]	PD3ON	Power Domain 3 Status bit This bit shows current power domain 3 status. '0': Power domain 3 switched OFF '1': Power domain 3 switched ON
[0]	PD2ON	Power Domain 2 Status bit This bit shows current power domain 2 status. '0': Power domain 2 switched OFF '1': Power domain 2 switched ON

2.2.7.2 Clock Source Enable Status Register (SYSC_CKSRESTSR)

This register shows the source clock status.

Clock Source Enable Status Register (SYSC_CKSRESTSR)

Figure 2-48. Clock Source Enable Status Register (SYSC_CKSRESTSR)

SYSC_CKSRESTSR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	GFXPLLRDY	SSCGPLLRDY	MAINPLLRDY	SUBCLKRDY	MAINCLKRDY	RCCLKRDY	SRCCLKRDY	read0	GFXPLLEN	SSCGPLLEN	MAINPLLEN	SOSCEN	MOSCEN	RCOSCEN	SRCOSCEN
R0	R	R	R	R	R	R	R	R0	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Table 2-49. Clock Source Enable Status Register (SYSC_CKSRESTSR) bits

Bit Position	Bit Field Name	Bit Description
[15]	read0	-
[14]	GFXPLLRDY	GFX PLL Ready bit This bit shows GFX PLL oscillation circuit stabilization status. '0': GFX PLL oscillation not stabilized '1': GFX PLL oscillation stabilized
[13]	SSCGPLLRDY	SSCG PLL Ready bit This bit shows SSCG PLL oscillation circuit stabilization status. '0': SSCG PLL oscillation not stabilized '1': SSCG PLL oscillation stabilized
[12]	MAINPLLRDY	Main PLL Ready bit This bit shows Main PLL oscillation circuit stabilization status. '0': Main PLL oscillation not stabilized '1': Main PLL oscillation stabilized
[11]	SUBCLKRDY	Sub Oscillator Ready bit This bit shows Sub oscillation circuit stabilization status. '0': Sub oscillation not stabilized '1': Sub oscillation stabilized When this bit is '1' Source Clock Timer (SUBSCT) can be used as normal timer to count Sub clock.

Table 2-49. Clock Source Enable Status Register (SYSC_CKSRESTSR) bits

Bit Position	Bit Field Name	Bit Description
[10]	MAINCLKRDY	Main Oscillator Ready bit This bit shows Main oscillation circuit stabilization status. '0': Main oscillation not stabilized '1': Main oscillation stabilized When this bit is '1' Source Clock Timer (MAINSCT) can be used as normal timer to count Main clock.
[9]	RCCLKRDY	RC Oscillator Ready bit This bit shows RC oscillation circuit stabilization status. '0': RC oscillation not stabilized '1': RC oscillation stabilized When this bit is '1' Source Clock Timer (RCSCT) can be used as normal timer to count RC clock.
[8]	SRCCLKRDY	Slow RC Oscillator Ready bit This bit shows Slow RC oscillation circuit stabilization status. '0': Slow RC oscillator not stabilized '1': Slow RC oscillator stabilized When this bit is '1' Source Clock Timer (SRCSCT) can be used as normal timer to count Slow RC clock.
[7]	read0	-
[6]	GFXPLLEN	GFX PLL Enable Status bit This bit shows GFX PLL oscillation circuit status. '0': GFX PLL oscillator disabled '1': GFX PLL oscillator enabled
[5]	SSCGPLLEN	SSCG PLL Enable Status bit This bit shows SSCG PLL oscillation circuit status. '0': SSCG PLL oscillator disabled '1': SSCG PLL oscillator enabled
[4]	MAINPLLEN	Main PLL Enable Status bit This bit shows Main PLL oscillation circuit status. '0': Main PLL oscillator disabled '1': Main PLL oscillator enabled
[3]	SOSCEN	Sub Oscillator Enable Status bit This bit shows Sub oscillation circuit status. '0': Sub oscillator disabled '1': Sub oscillator enabled
[2]	MOSCEN	Main Oscillator Enable Status bit This bit shows Main oscillation circuit status. '0': Main oscillator disabled '1': Main oscillator enabled
[1]	RCOSCEN	RC Oscillator Enable Status bit This bit shows RC oscillation circuit status. '0': RC oscillator disabled '1': RC oscillator enabled

Table 2-49. Clock Source Enable Status Register (SYSC_CKSRESTSR) bits

Bit Position	Bit Field Name	Bit Description
[0]	SRCOSCEN	Slow RC Oscillator Enable Status bit This bit shows Slow RC oscillation circuit status. '0': Slow RC oscillator disabled '1': Slow RC oscillator enabled

2.2.7.3 Clock Select Status Register (SYSC_CKSELSTSR)

This register shows the clock selection multiplexer status.

Clock Select Status Register (SYSC_CKSELSTSR)

Figure 2-49. Clock Select Status Register (SYSC_CKSELSTSR)

SYSC_CKSELSTSR																														
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**
read0	read0	SPICSL[1]	SPICSL[0]	read0	read0	PIXCSL[1]	PIXCSL[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI3CSL[2]	PERI3CSL[1]	PERI3CSL[0]	read0	PERI1CSL[2]	PERI1CSL[1]	PERI1CSL[0]	read0	PERI0CSL[2]	PERI0CSL[1]	PERI0CSL[0]	read0	SYSCSL[2]	SYSCSL[1]	SYSCSL[0]
R0	R0	R	R	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R0	R	R	R	R0	R	R	R	R0	R	R	R
0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-50. Clock Select Status Register (SYSC_CKSELSTSR) bits

Bit Position	Bit Field Name	Bit Description
[31:30]	read0	-
[29:28]	SPICSL	SPI Clock Select Status for Graphics Subsystem These bits show clock selection status for the Graphics Subsystem SPI clock. '00': SPI clock (Main PLL clock) is selected '01': External clock is selected '10' to '11': Clock is tied low
[27:26]	read0	-
[25:24]	PIXCSL	Pixel Clock Select Status for Graphics Subsystem These bits show clock selection status for the Graphics Subsystem pixel clock. '00': GFX bus clock is selected '01': External clock is selected '10': GFX PLL clock is selected '11': Clock is tied low
[23:16]	read0	-
[15]	read0	-

Table 2-50. Clock Select Status Register (SYSC_CKSELSTSR) bits

Bit Position	Bit Field Name	Bit Description
[14:12]	PERI3CSL	Peripheral Group 3 Clock Select Status bit These bits show the clock selection status for the peripheral group 3. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low
[11]	read0	-
[10:8]	PERI1CSL	Peripheral Group 1 Clock Select Status bit These bits show the clock selection status for the peripheral group 1. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low
[7]	read0	-
[6:4]	PERI0CSL	Peripheral Group 0 Clock Select Status bit These bits show the clock selection status for the peripheral group 0. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low
[3]	read0	-
[2:0]	SYSCSL	System Clock Select Status bit These bits show the clock selection status for the core group. '000': RC clock is selected '001': Sub oscillator clock is selected '010': Main oscillator clock is selected '011': Main PLL clock is selected '100': SSCG PLL clock is selected '101' to '111': Clock is tied low

2.2.7.4 Clock Enable Status Register (SYSC_CKESTSR)

This register shows the clock gating status.

Clock Enable Status Register (SYSC_CKESTSR)

Figure 2-50. Clock Enable Status Register (SYSC_CKESTSR)

SYSC_CKESTSR																													
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**
read0	ENSPIPD5	ENPIXPD5	ENGFXPD5	read0	read0	read0	ENCFGPD4	read0	read0	ENEXTBUSPD3	ENSPIPD3	ENMEMEPD3	ENHPMPD3	ENTRACEPD3	ENSYSPD3	read0	read0	read0	read0	ENPERI4PD2	ENPERI3PD2	read0	ENPERI1PD2	ENPERI0PD2	ENDMAPD2	ENTRACEPD2	ENHPMPD2	read0	read0	read0	ENCFGPD1
R0	R	R	R	R0	R0	R0	R	R0	R0	R	R	R	R	R	R	R0	R0	R0	R0	R	R	R0	R	R	R	R	R	R0	R0	R0	R
0	1	1	1	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	1

Table 2-51. Clock Enable Status Register (SYSC_CKESTSR) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30]	ENSPIPD5	Graphics SPI Clock Status for Power Domain 5 This bit shows the SPI clock status to the Graphics Subsystem. '0': SPI clock to the Graphics Subsystem is disabled '1': SPI clock to the Graphics Subsystem is enabled
[29]	ENPIXPD5	Graphics PLL Clock Status for Power Domain 5 This bit shows the pixel clock status to the Graphics Subsystem. '0': Pixel clock to the Graphics Subsystem is disabled '1': Pixel clock to the Graphics Subsystem is enabled
[28]	ENGFXPD5	Graphics Bus Clock Status for Power Domain 5 This bit shows bus clock status to the Graphics Subsystem. '0': Bus clock to the Graphics Subsystem is disabled '1': Bus clock to the Graphics Subsystem is enabled
[27:25]	read0	-
[24]	ENCFGPD4	Configuration Clock Status for Power Domain 4 This bit shows clocks status to the retention RAM. '0': Retention RAM clock is disabled '1': Retention RAM clock is enabled
[23:22]	read0	-
[21]	ENEXTBUSPD3	Enable External Bus Clock Status for Power Domain 3 This bit shows the current clock status to the external bus peripheral. '0': Clock to external bus peripheral disabled '1': Clock to external bus peripheral enabled

Table 2-51. Clock Enable Status Register (SYSC_CKESTSR) bits

Bit Position	Bit Field Name	Bit Description
[20]	ENSPIPD3	HSSPI SPI Clock Status for Power Domain 3 This bit shows SPI clock status to HSSPI module. '0': SPI clock of HSSPI is disabled '1': SPI clock of HSSPI is enabled
[19]	ENMEMEPD3	Memory External Clock Status for Power Domain 3 This bit shows the current clock status to the external memory group. '0': Clock to external memory group disabled '1': Clock to external memory group enabled
[18]	ENHPMPD3	HPM Clock for Status Power Domain 3 This bit shows the current clock status to the AXI port of TCFLASH interface. '0': Clock to the AXI port of TCFLASH interface disabled '1': Clock to the AXI port of TCFLASH interface enabled
[17]	ENTRACEPD3	Trace Clock Status for Power Domain 3 This bit shows the current clock status to the debug and trace group in power domain 3. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled
[16]	ENSYSPD3	System Clock Status for Power Domain 3 This bit shows the current clock status to the core group. '0': Clock to core group disabled '1': Clock to core group enabled
[15:12]	read0	-
[11]	ENPERI4PD2	Peripheral Group 4 Clock Status for Power Domain 2 This bit shows the current clock status to the peripheral group 4. '0': Clock to peripheral group 4 disabled '1': Clock to peripheral group 4 enabled
[10]	ENPERI3PD2	Peripheral Group 3 Clock Status for Power Domain 2 This bit shows the current clock status to the peripheral group 3. '0': Clock to peripheral group 3 disabled '1': Clock to peripheral group 3 enabled
[9]	read0	-
[8]	ENPERI1PD2	Peripheral Group 1 Clock Status for Power Domain 2 This bit shows the current clock status to the peripheral group 1. '0': Clock to peripheral group 1 disabled '1': Clock to peripheral group 1 enabled
[7]	ENPERI0PD2	Peripheral Group 0 Clock Status for Power Domain 2 This bit shows the current clock status to the peripheral group 0. '0': Clock to peripheral group 0 disabled '1': Clock to peripheral group 0 enabled

Table 2-51. Clock Enable Status Register (SYSC_CKESTSR) bits

Bit Position	Bit Field Name	Bit Description
[6]	ENDMAPD2	DMA Clock Status for Power Domain 2 This bit shows the current clock status to the DMA in power domain 2. '0': Clock to DMA disabled '1': Clock to DMA enabled
[5]	ENTRACEPD2	Trace Clock Status for Power Domain 2 This bit shows the current clock status to the debug and trace group in power domain 2. '0': Clock to debug and trace group disabled '1': Clock to debug and trace group enabled
[4]	ENHPMPD2	HPM Clock Status for Power Domain 2 This bit shows the current clock status to the HPM bus matrix. '0': Clock to HPM bus matrix disabled '1': Clock to HPM bus matrix enabled
[3:1]	read0	-
[0]	ENCFGPD1	Configuration Clock Status for Power Domain 1 This bit shows the current clock status to the configuration group. '0': Clock to configuration group disabled '1': Clock to configuration group enabled

2.2.7.5 Clock Divider Register 0 (SYSC_CKDIVSTSR0)

This register shows the clock division ratio status.

Clock Divider Register 0 (SYSC_CKDIVSTSR0)

Figure 2-51. Clock Divider Register 0 (SYSC_CKDIVSTSR0)

SYSC_CKDIVSTSR0																															
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**
read0	read0	read0	read0	read0	read0	CFGDIV[1]	CFGDIV[0]	read0	read0	read0	read0	read0	read0	HPMDIV[1]	HPMDIV[0]	read0	read0	read0	read0	read0	read0	TRACEDIV[1]	TRACEDIV[0]	read0	read0	read0	read0	SYSDIV[3]	SYSDIV[2]	SYSDIV[1]	SYSDIV[0]
R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-52. Clock Divider Register 0 (SYSC_CKDIVSTSR0) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	CFGDIV	Configuration Clock Division Value These bits show current clock divider settings for the configuration group clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	HPMDIV	HPM Clock Division Value These bits show current clock divider settings for the HPM clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-
[9:8]	TRACEDIV	Trace Clock Division Value These bits show current clock divider settings for the debug and trace clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8

Table 2-52. Clock Divider Register 0 (SYSC_CKDIVSTSR0) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3:0]	SYSDIV	System Clock Division Value These bits show current clock divider settings for the system clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.7.6 Clock Divider Register 1 (SYSC_CKDIVSTSR1)

This register shows the clock division ratio status.

Clock Divider Register 1 (SYSC_CKDIVSTSR1)

Figure 2-52. Clock Divider Register 1 (SYSC_CKDIVSTSR1)

SYSC_CKDIVSTSR1																															
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**
read0	read0	read0	read0	read0	read0	GFXDIV[1]	GFXDIV[0]	read0	read0	read0	read0	read0	read0	PERI4DIV[1]	PERI4DIV[0]	read0	read0	read0	read0	read0	read0	MEMEDIV[1]	MEMEDIV[0]	read0	read0	read0	read0	read0	EXTBUSDIV[2]	EXTBUSDIV[1]	EXTBUSDIV[0]
R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-53. Clock Divider Register 1 (SYSC_CKDIVSTSR1) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	GFXDIV	Graphics Clock Division Value These bits show current clock divider settings for the Graphics Sub-system bus clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[23:18]	read0	-
[17:16]	PERI4DIV	Peripheral 4 Clock Division Value These bits show the current clock divider for the peripheral group 4 clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[15:10]	read0	-

Table 2-53. Clock Divider Register 1 (SYSC_CKDIVSTSR1) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	MEMEDIV	Memory External Clock Division Value These bits show the current clock divider for the external memory group (HSSPI module) clock. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 8
[7:3]	read0	-
[2:0]	EXTBUSDIV	External Bus Clock Division Value These bits show the current clock divider for the external bus clock. For more details refer to Figure 2-132. '000': Clock division is bypassed (division by 1) '001': Divided by 2 '010': Divided by 4 '011': Divided by 8 .. '110': Divided by 64, '111': Divided by 128,

2.2.7.7 Clock Divider Register 2 (SYSC_CKDIVSTSR2)

This register shows the clock division ratio status.

Clock Divider Register 2 (SYSC_CKDIVSTSR2)

Figure 2-53. Clock Divider Register 2 (SYSC_CKDIVSTSR2)

SYSC_CKDIVSTSR2																															
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**
read0	read0	read0	read0	PERI3DIV[3]	PERI3DIV[2]	PERI3DIV[1]	PERI3DIV[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PERI1DIV[3]	PERI1DIV[2]	PERI1DIV[1]	PERI1DIV[0]	read0	read0	read0	read0	PERI0DIV[3]	PERI0DIV[2]	PERI0DIV[1]	PERI0DIV[0]
R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-54. Clock Divider Register 2 (SYSC_CKDIVSTSR2) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:24]	PERI3DIV	Peripheral Group 3 Clock Division Value These bits show current clock divider settings for the peripheral group 3. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[23:16]	read0	-
[15:12]	read0	-
[11:8]	PERI1DIV	Peripheral Group 1 Clock Division Value These bits show current clock divider settings for the peripheral group 1. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:4]	read0	-

Table 2-54. Clock Divider Register 2 (SYSC_CKDIVSTSR2) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	PERI0DIV	Peripheral Group 0 Clock Division Value These bits show current clock divider settings for the peripheral group 0. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30

2.2.7.8 PLL Status Register (SYSC_PLLSTSR)

This register shows the Main PLL configuration state. For PLL stabilization time settings refer to 2.3.5.3 Source clocks.

PLL Status Register (SYSC_PLLSTSR)

Figure 2-54. PLL Status Register (SYSC_PLLSTSR)

SYSC_PLLSTSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	PLLDIVN[6]	PLLDIVN[5]	PLLDIVN[4]	PLLDIVN[3]	PLLDIVN[2]	PLLDIVN[1]	PLLDIVN[0]	read0	read0	read0	read0	PLLDIVM[3]	PLLDIVM[2]	PLLDIVM[1]	PLLDIVM[0]	read0	read0	read0	read0	read0	read0	PLLDIVL[1]	PLLDIVL[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-55. PLL Status Register (SYSC_PLLSTSR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	read0	-
[22:16]	PLLDIVN	PLL Multiplication-by-N Value These bits show current multiplication value for Main PLL clock. '0000000': PLL input clock is multiplied by 1 '0000001': PLL input clock is multiplied by 2 ... '1111111': PLL input clock is multiplied by 128 Refer to device specific datasheet for Main PLL output frequency range.
[15:12]	read0	-
[11:8]	PLLDIVM	PLL Division-by-M Value These bits show current clock divider settings for Main PLL output clock. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:2]	read0	-

Table 2-55. PLL Status Register (SYSC_PLLSTSR) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	PLLDIVL	PLL Input Division Value These bits show the current clock divider for the Main PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.7.9 SSCG PLL Status Register 0 (SYSC_SSCGSTSR0)

This register shows the SSCG PLL configuration state. For SSCG-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

SSCG PLL Status Register 0 (SYSC_SSCGSTSR0)

Figure 2-55. SSCG PLL Status Register 0 (SYSC_SSCGSTSR0)

SYSC_SSCGSTSR0																															
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**
read0	read0	read0	SSCGDIVP[4]	SSCGDIVP[3]	SSCGDIVP[2]	SSCGDIVP[1]	SSCGDIVP[0]	read0	read0	SSCGDIVN[5]	SSCGDIVN[4]	SSCGDIVN[3]	SSCGDIVN[2]	SSCGDIVN[1]	SSCGDIVN[0]	read0	read0	read0	read0	SSCGDIVM[3]	SSCGDIVM[2]	SSCGDIVM[1]	SSCGDIVM[0]	read0	read0	read0	read0	read0	read0	SSCGDIVL[1]	SSCGDIVL[0]
R0	R0	R0	R	R	R	R	R	R0	R0	R	R	R	R	R	R	R0	R0	R0	R0	R	R	R	R	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 2-56. SSCG PLL Status Register 0 (SYSC_SSCGSTSR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	SSCGDIVP	SSCG PLL Multiplication-by-P Value These bits shows the current SSCGDIVP setting, which along with SYSC_SSCGSTSR0:SSCGDIVN bits define the multiplication value settings for SSCG PLL clock. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_SSCGSTSR0:SSCGDIVN x 1) '00010':PLL input clock multiplied by (SYSC_SSCGSTSR0:SSCGDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_SSCGSTSR0:SSCGDIVN x 31)
[23:22]	read0	-

Table 2-56. SSCG PLL Status Register 0 (SYSC_SSCGSTSR0) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	SSCGDIVN	SSCG PLL Multiplication-by-N Value These bits shows the current SSCGDIVN setting, which along with SYSC_SSCGSTSR0:SSCGDIVP bits define the multiplication value for SSCG PLL clock. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_SSCGSTSR0:SSC-GDIVP x 2) '000011': PLL input clock multiplied by (SYSC_SSCGSTSR0:SSC-GDIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_SSCGSTSR0:SSC-GDIVP x 63)
[15:12]	read0	-
[11:8]	SSCGDIVM	SSCG PLL Division-by-M Value These bits show current clock divider settings for the division value for SSCG PLL. For more details refer to Figure 2-132. '0000': Clock division is bypassed (division by 1) '0001': Divided by 2 '0010': Divided by 4 '0011': Divided by 6 ... '1111': Divided by 30
[7:2]	read0	-
[1:0]	SSCGDIVL	SSCG PLL Input Division Value These bits show the current clock divider for the SSCG PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.7.10 SSCG PLL Status Register 1 (SYSC_SSCGSTSR1)

This register shows the SSCG PLL configuration state.

SSCG PLL Status Register 1 (SYSC_SSCGSTSR1)

Figure 2-56. SSCG PLL Status Register 1 (SYSC_SSCGSTSR1)

SYSC_SSCGSTSR1																															
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**
read0	read0	read0	read0	read0	read0	read0	SSCGSSEN	read0	read0	read0	read0	read0	SSCGFREQ[1]	SSCGFREQ[0]	SSCGMODE	read0	read0	read0	read0	read0	read0	SSCGRATE[9]	SSCGRATE[8]	SSCGRATE[7]	SSCGRATE[6]	SSCGRATE[5]	SSCGRATE[4]	SSCGRATE[3]	SSCGRATE[2]	SSCGRATE[1]	SSCGRATE[0]
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-57. SSCG PLL Status Register 1 (SYSC_SSCGSTSR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	SSCGSSEN	SSCG PLL Modulation Enable Status These bits show the current modulation setting for SSCG PLL. '0': SSCG PLL modulation is disabled '1': SSCG PLL modulation is enabled
[23:19]	read0	-
[18:17]	SSCGFREQ	SSCG PLL Modulation Frequency Select These bits show the current modulation frequency rate settings for SSCG PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	SSCGMODE	SSCG PLL Modulator Mode This bit shows current modulation mode settings of SSCG PLL. '0': SSCG PLL works in down spread mode '1': SSCG PLL works in center spread mode
[15:10]	read0	-

Table 2-57. SSCG PLL Status Register 1 (SYSC_SSCGSTSR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	SSCGRATE	SSCG PLL Modulation Rate Status Value These bits show the current modulation rate settings for SSCG PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.7.11 Graphics PLL Status Register 0 (SYSC_GFXSTSR0)

This register shows the Graphics PLL configuration status. For GFX-PLL stabilization time settings refer to 2.3.5.3 Source clocks.

Graphics PLL Status Register 0 (SYSC_GFXSTSR0)

Figure 2-57. Graphics PLL Status Register 0 (SYSC_GFXSTSR0)

SYSC_GFXSTSR0																															
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**
read0	read0	read0	GFXDIVP[4]	GFXDIVP[3]	GFXDIVP[2]	GFXDIVP[1]	GFXDIVP[0]	read0	read0	GFXDIVN[5]	GFXDIVN[4]	GFXDIVN[3]	GFXDIVN[2]	GFXDIVN[1]	GFXDIVN[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GFXDIVL[1]	GFXDIVL[0]
R0	R0	R0	R	R	R	R	R	R0	R0	R	R	R	R	R	R	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-58. Graphics PLL Status Register 0 (SYSC_GFXSTSR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	GFXDIVP	Graphics PLL Division-by-P Value These bits shows the current GFXDIVP setting, which along with SYSC_GFXSTSR0:GFXDIVN bits define the multiplication value settings for Graphics PLL clock. '00000':Reserved '00001':PLL input clock multiplied by (SYSC_GFXSTSR0:GFXDIVN x 1) '00010':PLL input clock multiplied by (SYSC_GFXSTSR0:GFXDIVN x 2) ... '11111':PLL input clock multiplied by (SYSC_GFXSTSR0:GFXDIVN x 31)
[23:22]	read0	-

Table 2-58. Graphics PLL Status Register 0 (SYSC_GFXSTSR0) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	GFXDIVN	Graphics PLL Division-by-N Value These bits show the current GFXDIVN setting, which along with SYSC_GFXSTSR0:GFXDIVP bits define the multiplication value for Graphics PLL clock. '000000' to '000001': Reserved '000010': PLL input clock multiplied by (SYSC_GFXSTSR0:GFX-DIVP x 2) '000011': PLL input clock multiplied by (SYSC_GFXSTSR0:GFX-DIVP x 3) ... '111111': PLL input clock multiplied by (SYSC_GFXSTSR0:GFX-DIVP x 63)
[15:8]	read0	-
[7:2]	read0	-
[1:0]	GFXDIVL	Graphics PLL Input Division Value These bits show the current clock divider for the GFX PLL macro. For more details refer to Figure 2-132. '00': Clock division is bypassed (division by 1) '01': Divided by 2 '10': Divided by 4 '11': Divided by 6

2.2.7.12 Graphics PLL Status Register (SYSC_GFXSTSR1)

This register shows the Graphics PLL configuration status.

Graphics PLL Status Register (SYSC_GFXSTSR1)

Figure 2-58. Graphics PLL Status Register 1 (SYSC_GFXSTSR1)

SYSC_GFXSTSR1																															
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**
read0	read0	read0	read0	read0	read0	read0	GFXSSEN	read0	read0	read0	read0	read0	GFXFREQ[1]	GFXFREQ[0]	GFXMODE	read0	read0	read0	read0	read0	read0	GFXRATE[9]	GFXRATE[8]	GFXRATE[7]	GFXRATE[6]	GFXRATE[5]	GFXRATE[4]	GFXRATE[3]	GFXRATE[2]	GFXRATE[1]	GFXRATE[0]
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1

Table 2-59. Graphics PLL Status Register 1 (SYSC_GFXSTSR1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	GFXSSEN	GFX PLL Modulation Enable Status These bits show the current modulation setting for GFX PLL. '0': GFX PLL modulation is disabled '1': GFX PLL modulation is enabled
[23:19]	read0	-
[18:17]	GFXFREQ	Graphics PLL Modulation Frequency Select These bits show the current modulation frequency rate settings for GFX PLL. '00': [Fmod = (1/1024) x Fin] '01': [Fmod = (1/2048) x Fin] '10': [Fmod = (1/4096) x Fin] '11': [Fmod = (1/4096) x Fin]
[16]	GFXMODE	Graphics PLL Modulator Mode This bit shows the current modulation mode settings of GFX PLL. '0': GFX PLL works in down spread mode '1': GFX PLL works in center spread mode
[15:10]	read0	-

Table 2-59. Graphics PLL Status Register 1 (SYSC_GFXSTSR1) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	GFXRATE	Graphics PLL Modulation Rate Status Value These bits show the current modulation rate settings for GFX PLL for EMI reduction. '0000101001': 0.5% modulation rate '0001010010': 1.0% modulation rate '0010100100': 2.0% modulation rate '0011110110': 3.0% modulation rate '0101001000': 4.0% modulation rate '0110011010': 5.0% modulation rate Others: Reserved Refer to Table 2-113 for details on limitation of modulation rate and Divider-N settings.

2.2.7.13 LVD Configuration Status Register (SYSC_LVDCFGSTSR)

This register shows the LVD configuration status.

LVD Configuration Status Register (SYSC_LVDCFGSTSR)

Figure 2-59. LVD Configuration Status Register (SYSC_LVDCFGSTSR)

SYSC_LVDCFGSTSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	LVDRDY12	SV12[2]	SV12[1]	SV12[0]	LVDR12	LVDE12	read0	read0	LVDRDY33	SV33[2]	SV33[1]	SV33[0]	LVDR33	LVDE33	read0	read0	LVDRDY50	SV50[2]	SV50[1]	SV50[0]	LVDR50	LVDE50
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R0	R0	R	R	R	R	R	R	R0	R0	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	1	1	1

Table 2-60. LVD Configuration Status Register (SYSC_LVDCFGSTSR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:22]	read0	-
[21]	LVDRDY12	Low Voltage Detector Ready Status for 1.2 V This bit shows the status of LVD stabilization. '0': LVD for 1.2 V is not functionally ready '1': LVD for 1.2 V is functionally ready LVD for 1.2 V generates reset in case the 1.2 V supply reduces below threshold voltage level.
[20:18]	SV12	Reference Voltage Selection for 1.2 V LVD These bits show the current analog threshold for Low Voltage Detection on 1.2 V core supply. '000': 0.5 V '001': 0.6 V '011': 0.7 V '010': 0.8 V '110': 0.9 V '111': 1.0 V '101': 1.1 V '100': 1.2 V On reset, 0.8 V is chosen as reference voltage.
[17]	LVDR12	Low Voltage Detector Reset/Interrupt Select Status for 1.2 V This bit shows the behaviour of LVD on 1.2 V core supply. '0': Low Voltage Detect on 1.2 V sets interrupt flag SYSC_SYSERRR:LVD12IF '1': Low Voltage Detect on 1.2 V generates reset

Table 2-60. LVD Configuration Status Register (SYSC_LVDCFGSTSR) bits

Bit Position	Bit Field Name	Bit Description
[16]	LVDE12	Low Voltage Detector Enable for 1.2 V This bit shows the current LVD functionality on 1.2 V core supply. '0': LVD on 1.2 V disabled '1': LVD on 1.2 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.
[15:14]	read0	-
[13]	LVDRDY33	Low Voltage Detector Ready Status for 3.3 V This bit shows the status of LVD stabilization. '0': LVD for 3.3 V is not functionally ready '1': LVD for 3.3 V is functionally ready LVD for 3.3 V will generate reset in case the 3.3 V supply reduces below threshold voltage level.
[12:10]	SV33	Reference Voltage Selection for 3.3 V LVD These bits show the current analog threshold for Low Voltage Detection on 3.3 V Graphics IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V On reset, 2.6 V is chosen as reference voltage.
[9]	LVDR33	Low Voltage Detector Reset/Interrupt Select Status for 3.3 V This bit shows the behaviour of LVD on 3.3 V Graphics IO supply. '0': Low Voltage Detect on 3.3 V sets interrupt flag SYSC_SYSERRR:LVD33IF '1': Low Voltage Detect on 3.3 V generates reset
[8]	LVDE33	Low Voltage Detector Enable for 3.3 V This bit shows the current LVD functionality on 3.3 V Graphics IO supply. '0': LVD on 3.3 V disabled '1': LVD on 3.3 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.
[7:6]	read0	-
[5]	LVDRDY50	Low Voltage Detector Ready Status for 5.0 V This bit shows the status of LVD stabilization. '0': LVD for 5.0 V is not functionally ready '1': LVD for 5.0 V is functionally ready LVD for 5.0 V will generate reset in case the 5.0 V supply reduces below threshold voltage level.

Table 2-60. LVD Configuration Status Register (SYSC_LVDCFGSTSR) bits

Bit Position	Bit Field Name	Bit Description
[4:2]	SV50	Reference Voltage Selection for 5.0 V LVD These bits show the current analog threshold for Low Voltage Detection on 5.0 V IO supply. '000': 2.2 V '001': 2.4 V '011': 2.6 V '010': 2.7 V '110': 3.7 V '111': 3.9 V '101': 4.1 V '100': 4.3 V On reset, 2.6 V is chosen as reference voltage.
[1]	LVDR50	Low Voltage Detector Reset/Interrupt Select Status for 5.0 V This bit shows the behavior of LVD on 5.0 V IO supply. '0': Low Voltage Detect on 5.0 V sets interrupt flag SYSC_SYSERRR:LVD50IF '1': Low Voltage Detect on 5.0 V generates reset
[0]	LVDE50	Low Voltage Detector Enable for 5.0 V This bit shows the current LVD functionality on 5.0 V IO supply. '0': LVD on 5.0 V disabled '1': LVD on 5.0 V enabled After enabling LVD, LVD functionality is gated till LVD stabilization time is over.

2.2.7.14 CSV Configuration Status Registe (SYSC_CSVCFGSTSR)

This register shows the CSV configuration status.

CSV Configuration Status Register (SYSC_CSVCFGSTSR)

Figure 2-60. CSV Configuration Status Register (SYSC_CSVCFGSTSR)

SYSC_CSVCFGSTSR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GPCSVE	SPCSVE	MPCSVE	SOCSVE	MOCSVE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-61. CSV Configuration Status Register (SYSC_CSVCFGSTSR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:5]	read0	-
[4]	GPCSVE	Graphics PLL Clock Supervisor Status This bit shows current clock supervision settings on Graphics PLL clock. '0': Graphics PLL clock supervision disabled '1': Graphics PLL clock supervision enabled
[3]	SPCSVE	SSCG PLL Clock Supervisor Status This bit shows current clock supervision settings on SSCG PLL clock. '0': SSCG PLL clock supervision disabled '1': SSCG PLL clock supervision enabled
[2]	MPCSVE	Main PLL Clock Supervisor Status This bit shows current clock supervision settings on Main PLL clock. '0': Main PLL clock supervision disabled '1': Main PLL clock supervision enabled
[1]	SOCSVE	Sub Oscillator Clock Supervisor Status This bit shows clock supervision status on Sub clock. '0': Sub clock supervision disabled '1': Sub clock supervision enabled
[0]	MOCSVE	Main Oscillator Clock Supervisor Status This bit shows clock supervision status on Main clock. '0': Main clock supervision disabled '1': Main clock supervision enabled

2.2.8 System registers

System registers have system ID and interrupt related control and status bits. This register gets initialized on hard resets.

2.2.8.1 System ID Register (SYSC_SYSIDR)

System Identification Register (SYSC_SYSIDR) shows the 32-bit version number of the chip. This is a read-only register.

System Identification Register (SYSC_SYSIDR)

Figure 2-61. System Identification Register (SYSC_SYSIDR)

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

Table 2-62. System Identification Register (SYSC_SYSIDR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CHIPID	Chip Identification Number These 32-bit value indicates the version number of the chip. For more details refer device specific datasheet.

2.2.8.2 System Status Register (SYSC_SYSSTSR)

System Status Register (SYSC_SYSSTSR) shows status of PSS and RUN profiles. It gives the validity status of the profile register setting. It also tells if profile is applied or being applied. None of the bits in the System Status Register (SYSC_SYSSTSR) causes an interrupt (neither IRQ nor NMI, except bit SYSC_SYSSTSR:RUNDN which generates IRQ).

System Status Register (SYSC_SYSSTSR)

Figure 2-62. System Status Register (SYSC_SYSSTSR)

SYSC_SYSSTSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PSSBUSY	RUNBUSY	read0	read0	read0	read0	read0	DEVSTAT	PSSDN	RUNDN	read0	read0	read0	read0	read0	IPPAPSS	IRPAPSS	IRPARUN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Table 2-63. System Status Register (SYSC_SYSSTSR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:18]	read0	-
[17]	PSSBUSY	PSS Profile Applying Status This bit shows if PSS setting is being applied and state controller is busy. '0': No PSS setting is being applied. State controller is idle '1': New PSS profile settings are being applied. State controller is busy
[16]	RUNBUSY	RUN Profile Applying Status This bit shows if RUN setting is being applied and state controller is busy. '0': No RUN setting is being applied. State controller is idle '1': New RUN profile settings are being applied. State controller is busy
[15:11]	read0	-
[10]	DEVSTAT	Device State Status This bit shows the current state of device. '0': Device state is PSS '1': Device state is RUN

Table 2-63. System Status Register (SYSC_SYSSTSR) bits

Bit Position	Bit Field Name	Bit Description
[9]	PSSDN	PSS Profile Done This bit shows the status of last applied PSS profile. '0': Last applied PSS profile settings getting applied or not triggered for new settings '1': Last applied PSS profile settings applied on System Controller This bit is cleared by writing '1' in SYSC_SYSICLR:PSSDNICLR register bit.
[8]	RUNDN	RUN Profile Done This bit shows the status of last applied RUN profile. '0': Last applied RUN profile settings getting applied or not triggered for new settings '1': Last applied RUN profile settings applied on System Controller This bit is used for interrupt generation if SYSC_SYSINTER:RUNDNIE = '1'. This bit is cleared by writing '1' in SYSC_SYSICLR:RUNDNICLR register bit.
[7:3]	read0	-
[2]	IPPAPSS	Invalid PSS Profile for PSS Switching This bit shows validity of current configured PSS profile settings for PSS switching. '0': PSS profile register settings are valid for PSS switching '1': PSS profile register settings are invalid for PSS switching Before PSS state switching user should check the validity of settings. If settings for PSS profile registers are incorrect SYSC_SYSERRR:PSSERRIF bit will be set immediately after 0xBA is written to SYSC_PSSENR:PSSEN. **Note:** This is evaluated on the fly without dependency on SYSC_PSSENR:PSSEN.
[1]	IRPAPSS	Invalid RUN Profile for PSS Switching This bit shows validity of current configured RUN profile settings to be applied after device wake up from PSS state. '0': RUN profile register settings are valid for PSS switching '1': RUN profile register settings are invalid for PSS switching Before PSS state switching user should check the validity of RUN profile settings. If settings for RUN profile register for PSS switching are incorrect SYSC_SYSERRR:RUNWKERRIF bit will be set immediately after 0xBA is written to SYSC_PSSENR:PSSEN. IRPAPSS bit has some extra validity requirement (ex:CPU should use RC clock, LVD should be ON) for RUN profile if PD2 is switched OFF in PSS profile. **Note:** This is evaluated on fly without dependency on SYSC_PSSENR:PSSEN.

Table 2-63. System Status Register (SYSC_SYSSTSR) bits

Bit Position	Bit Field Name	Bit Description
[0]	IRPARUN	Invalid RUN Profile for Apply RUN This bit shows validity of current configured RUN profile settings for RUN switching. '0': RUN profile register settings are valid for RUN switching '1': RUN profile register settings are invalid for RUN switching Before applying any profile trigger user should check the validity of settings. **Note:** This is evaluated on the fly without dependency on SYSC_TRGRUNCNTR.

2.2.8.3 System Status Interrupt Enable Register (SYSC_SYSINTER)

System Status Interrupt Enable Register (SYSC_SYSINTER) controls the enable for RUN profile done interrupt.

System Status Interrupt Enable Register (SYSC_SYSINTER)

Figure 2-63. System Status Interrupt Enable Register (SYSC_SYSINTER)

SYSC_SYSINTER																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RUNDNIE	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-64. System Status Interrupt Enable Register (SYSC_SYSINTER) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	RUNDNIE	RUN Profile Applied Interrupt Enable bit '0': No interrupt is generated if SYSC_SYSSTSR:RUNDN bit sets '1': Interrupt is generated once RUN profile settings are applied successfully (when SYSC_SYSSTSR:RUNDN bit sets) **Note:** On return from PSS (Wakeup with reset), the BOOT ROM may clear SYSC_SYSINTER_RUNDNIE (and overrite the user setting)
[7:0]	read0	-

2.2.8.4 System Status Interrupt Clear Register (SYSC_SYSICLR)

System Status Interrupt Clear Register (SYSC_SYSICLR) can be used to clear the profile done flags.

System Status Interrupt Clear Register (SYSC_SYSICLR)

Figure 2-64. System Status Interrupt Clear Register (SYSC_SYSICLR)

SYSC_SYSICLR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PSSDNICLR	RUNDNICLR	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WPS1	R0WPS1	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-65. System Status Interrupt Clear Register (SYSC_SYSICLR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	PSSDNICLR	PSS Profile Applied Clear bit Writing '1' in this register bit clears SYSC_SYSSTSR:PSSDN flag. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[8]	RUNDNICLR	RUN Profile Applied Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSSTSR:RUNDN flag. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[7:0]	read0	-

Note: PSS profile done does not generate interrupt, since it causes the wakeup of the device.

2.2.8.5 System Error Register (SYSC_SYSERRR)

System Error Register (SYSC_SYSERRR) shows the status of profile error and Clock Supervisor error. Each of the register bits in System Error Register (SYSC_SYSERRR) causes NMI.

System Error Register (SYSC_SYSERRR)

Figure 2-65. System Error Register (SYSC_SYSERRR)

SYSC_SYSERRR																															
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**
read0	read0	read0	read0	read0	LVD12IF	LVD33IF	LVD50IF	read0	read0	PSSENEIF	RUNTRGEIF	read0	read0	read0	TRGERRIF	read0	read0	read0	GPMISS	SPMISS	MPMISS	SOMISS	MOMISS	read0	read0	read0	read0	read0	PSSERRIF	RUNWKERRIF	RUNERRIF
R0	R0	R0	R0	R0	R	R	R	R0	R0	R	R	R0	R0	R0	R	R0	R0	R0	R	R	R	R	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-66. System Error Register (SYSC_SYSERRR) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26]	LVD12IF	Low Voltage Detect for 1.2 V Interrupt Flag bit This bit is set when 1.2 V core voltage falls below the threshold value configured in SYSC_APPLVDCFGR:SV12[2:0] of Applied Profile Register. This bit is cleared by writing '1' in SYSC_SYSERRICLR:LVD12ICLR register bit.
[25]	LVD33IF	Low Voltage Detect for 3.3 V Interrupt Flag bit This bit is set when 3.3 V graphics IO voltage falls below the threshold value configured in SYSC_APPLVDCFGR:SV33[2:0] of Applied Profile Register. This bit is cleared by writing '1' in SYSC_SYSERRICLR:LVD33ICLR register bit.
[24]	LVD50IF	Low Voltage Detect for 5.0 V Interrupt Flag bit This bit is set when 5.0 V IO voltage falls below the threshold value configured in SYSC_APPLVDCFGR:SV50[2:0] of Applied Profile Register. This bit is cleared by writing '1' in SYSC_SYSERRICLR:LVD50ICLR register bit.
[23:22]	read0	-

Table 2-66. System Error Register (SYSC_SYSERRR) bits

Bit Position	Bit Field Name	Bit Description
[21]	PSSENEIF	PSS Profile Enable Error Interrupt Flag bit This bit sets when the PSS Enable Register (SYSC_PSSENR:PSSEN) is written with wrong enable value ie, written with value other than 0xBA. This bit is cleared by writing '1' in SYSC_SYSERRICLR:PSSENEICLR register bit.
[20]	RUNTRGEIF	RUN Profile Trigger Error Interrupt Flag bit This bit sets when the Trigger RUN Control Register (SYSC_TRGRUNCNTR:APPLYRUN) is written with wrong trigger value ie, written with value other than 0xAB. This bit is cleared by writing '1' in SYSC_SYSERRICLR:RUNTRGEICLR register bit.
[19:17]	read0	-
[16]	TRGERRIF	Trigger Error Interrupt Flag bit This bit is set when the current profile is about to be applied and the device state logic is triggered again for RUN profile settings. A new profile should not be applied before the previous state changes are completed. This bit is cleared by writing '1' in SYSC_SYSERRICLR:TRGERRICLR register bit.
[15:13]	read0	-
[12]	GPMISS	Graphics PLL Clock Missing Flag bit Graphics PLL oscillator clock missing interrupt status. '0': Graphics PLL clock miss not detected, Graphics PLL source is disabled or Graphics PLL CSV is disabled '1': Graphics PLL clock miss detected by Clock Supervisor This bit is cleared by writing '1' in SYSC_SYSERRICLR:GPMISSICLR register bit.
[11]	SPMISS	SSCG PLL Clock Missing Flag bit SSCG PLL oscillator clock missing interrupt status. '0': SSCG PLL clock miss not detected, SSCG PLL source is disabled or SSCG PLL CSV is disabled '1': SSCG PLL clock miss detected by Clock Supervisor This bit is cleared by writing '1' in SYSC_SYSERRICLR:SPMISSICLR register bit.
[10]	MPMISS	Main PLL Clock Missing Flag bit Main PLL oscillator clock missing interrupt status. '0': Main PLL clock miss not detected, Main PLL source is disabled or Main PLL CSV is disabled '1': Main PLL clock miss detected by Clock Supervisor This bit is cleared by writing '1' in SYSC_SYSERRICLR:MPMISSICLR register bit.

Table 2-66. System Error Register (SYSC_SYSERRR) bits

Bit Position	Bit Field Name	Bit Description
[9]	SOMISS	Sub Oscillator Clock Missing Interrupt Flag bit Sub oscillator clock missing status. '0': Sub clock miss not detected, Sub clock source is disabled or Sub CSV is disabled '1': Sub clock miss detected by Clock Supervisor This bit is cleared by writing '1' in SYSC_SYSERRICLR:SOMISSICLR register bit.
[8]	MOMISS	Main Oscillator Clock Missing Interrupt Flag bit Main oscillator clock missing status. '0': Main clock miss not detected, Main clock source is disabled or Main CSV is disabled '1': Main clock miss detected by Clock Supervisor This bit is cleared by writing '1' in SYSC_SYSERRICLR:MOMISSICLR register bit.
[7:3]	read0	-
[2]	PSSERRIF	PSS Profile Error Interrupt Flag bit This bit sets when SYSC_SYSSTSR:IPPAPSS = '1' and either SYSC_PSSENR:PSSEN is set or PSS state is applied. This bit is cleared by writing '1' in SYSC_SYSERRICLR:PSSERRICLR register bit.
[1]	RUNWKERRIF	RUN Profile Error for Wakeup Interrupt Flag bit This bit sets when SYSC_SYSSTSR:IRPAPSS = '1' and either SYSC_PSSENR:PSSEN is set or PSS state is applied. This bit is cleared by writing '1' in SYSC_SYSERRICLR:RUNWKERRICLR register bit.
[0]	RUNERRIF	RUN Profile Error Interrupt Flag bit This bit sets when SYSC_SYSSTSR:IRPARUN = '1' and RUN trigger is applied. This bit is cleared by writing '1' in SYSC_SYSERRICLR:RUNERRICLR register bit.

2.2.8.6 System Error Interrupt Clear Register (SYSC_SYSERRICLR)

System Error Interrupt Clear Register (SYSC_SYSERRICLR) can be used to clear the error interrupt flags.

System Error Interrupt Clear Register (SYSC_SYSERRICLR)

Figure 2-66. System Error Interrupt Clear Register (SYSC_SYSERRICLR)

SYSC_SYSERRICLR																													
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**
read0	read0	read0	read0	read0	LVD12ICLR	LVD33ICLR	LVD50ICLR	read0	read0	PSSENEICLR	RUNTRGEICLR	read0	read0	read0	TRGERRICLR	read0	read0	read0	GPMISSICLR	SPMISSICLR	MPMISSICLR	SOMISSICLR	MOMISSICLR	read0	read0	read0	read0	read0	PSSERRICLR	RUNWKERRICLR	RUNERRICLR
R0	R0	R0	R0	R0	R0WPS1	R0WPS1	R0WPS1	R0	R0	R0WPS1	R0WPS1	R0	R0	R0	R0WPS1	R0	R0	R0	R0WPS1	R0WPS1	R0WPS1	R0WPS1	R0WPS1	R0	R0	R0	R0	R0	R0WPS1	R0WPS1	R0WPS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-67. System Error Interrupt Clear Register (SYSC_SYSERRICLR) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26]	LVD12ICLR	LVD for 1.2 V Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:LVD12IF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[25]	LVD33ICLR	LVD for 3.3 V Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:LVD33IF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[24]	LVD50ICLR	LVD for 5.0 V Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:LVD50IF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[23:22]	read0	-

Table 2-67. System Error Interrupt Clear Register (SYSC_SYSERRICLR) bits

Bit Position	Bit Field Name	Bit Description
[21]	PSSENEICLR	PSS Enable Error Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:PSSENEIF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[20]	RUNTRGEICLR	RUN Trigger Error Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:RUNTRGEIF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[19:17]	read0	-
[16]	TRGERRICLR	Trigger Error Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:TRGERRIF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[15:13]	read0	-
[12]	GPMISSICLR	Graphics PLL Clock Missing Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:GPMISS flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[11]	SPMISSICLR	SSCG PLL Clock Missing Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:SPMISS flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[10]	MPMISSICLR	Main PLL Clock Missing Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:MPMISS flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.

Table 2-67. System Error Interrupt Clear Register (SYSC_SYSERRICLR) bits

Bit Position	Bit Field Name	Bit Description
[9]	SOMISSICLR	Sub Oscillator Clock Missing Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:SOMISS flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[8]	MOMISSICLR	Main Oscillator Clock Missing Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:MOMISS flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[7:3]	read0	-
[2]	PSSERRICLR	PSS Profile Error Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:PSSERRIF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[1]	RUNWKERRICLR	RUN Profile Error for Wakeup Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:RUNWKERRIF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.
[0]	RUNERRICLR	RUN Profile Error Interrupt Clear bit Writing '1' in this register bit clears SYSC_SYSERRR:RUNERRIF flag bit. Writing '0' in this register bit has no effect. This register bit is set by user and cleared automatically in the next cycle. Reading this bit always returns '0'.

2.2.9 Clock Supervisor Configuration registers

These registers contain values for Clock Supervisors threshold configuration and its test register bit to check the functioning of the CSVs. There are five Clock Supervisor modules for supervision of Main Clock, Sub Clock, Main PLL Clock, SSCG-PLL Clock, and Graphics PLL clock generators. CSV setting needs to be changed only when CSV is disabled. Writing CSV configuration when CSV is enabled generates PPU error. User should always program lower threshold value (CSVxx-CFGR:LOWTHR) greater than ‘1’.

2.2.9.1 Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR)

Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR) configures the reference counter window and the upper/lower threshold of the main counter for the Main oscillator clock supervision. Thr reference clock used by the Main oscillator CSV is the RC clock (CLK_RC_G).

Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR)

Figure 2-67. Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR)

SYSC_CSVMOCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	UPTHR[7]	UPTHR[6]	UPTHR[5]	UPTHR[4]	UPTHR[3]	UPTHR[2]	UPTHR[1]	UPTHR[0]	LOWTHR[7]	LOWTHR[6]	LOWTHR[5]	LOWTHR[4]	LOWTHR[3]	LOWTHR[2]	LOWTHR[1]	LOWTHR[0]	REFCLKWND[7]	REFCLKWND[6]	REFCLKWND[5]	REFCLKWND[4]	REFCLKWND[3]	REFCLKWND[2]	REFCLKWND[1]	REFCLKWND[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-68. Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	UPTHR	Upper Threshold Value Register bits These bits define the upper threshold of the counter at the end of each reference clock window. If the counter value is more than this threshold, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Table 2-68. Clock Supervisor Configuration for Main Clock Register (SYSC_CSVMOCFGR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	LOWTHR	Lower Threshold Value Register bits These bits define the lower threshold of the counter at the end of each reference clock window. If the counter value is lower than this threshold, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.
[7:0]	REFCLKWND	Reference Clock Window Value Register bits These bits define the duration in reference clocks for enabling the measurement of the clock to be monitored. After this time the monitoring clock counter will be compared with its lower and upper limits. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Note: Restrictions for the configuration of the UPTHR, LOWTHR and REFCLKWND bits in the Clock Supervisor Configuration Registers are given in 2.3.7.1 Block diagram.

2.2.9.2 Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR)

Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR) configures reference counter window and upper/lower threshold of Main counter for Sub oscillator clock supervision. Reference clock used by Sub oscillator CSV is Slow RC clock (CLK_SRC_G).

Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR)

Figure 2-68. Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR)

SYSC_CSVSOCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	UPTHR[7]	UPTHR[6]	UPTHR[5]	UPTHR[4]	UPTHR[3]	UPTHR[2]	UPTHR[1]	UPTHR[0]	LOWTHR[7]	LOWTHR[6]	LOWTHR[5]	LOWTHR[4]	LOWTHR[3]	LOWTHR[2]	LOWTHR[1]	LOWTHR[0]	REFCLKWND[7]	REFCLKWND[6]	REFCLKWND[5]	REFCLKWND[4]	REFCLKWND[3]	REFCLKWND[2]	REFCLKWND[1]	REFCLKWND[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-69. Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	UPTHR	Upper Threshold Value Register bits These bits define the upper threshold of the counter at the end of each reference clock window. If the counter value is more than this threshold, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.
[15:8]	LOWTHR	Lower Threshold Value Register bits These bits define the lower threshold of the counter at the end of each reference clock window. If the counter value is lower than this threshold, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Table 2-69. Clock Supervisor Configuration for Sub Clock Register (SYSC_CSVSOCFGR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	REFCLKWND	Reference Clock Window Value Register bits These bits define the duration in reference clocks for enabling the measurement of the clock to be monitored. After this time the monitoring clock counter will be compared with its lower and upper limits. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Note: Restrictions for the configuration of the UPTHR, LOWTHR and REFCLKWND bits in the Clock Supervisor Configuration Registers are given in 2.3.7.1 Block diagram.

2.2.9.3 Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR)

Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR) configures reference counter window and upper/lower threshold of main counter for Main PLL clock supervision. Reference clock used by Main PLL clock CSV is PLL input clock.

Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR)

Figure 2-69. Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR)

SYSC_CSVMPCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	UPTHR[7]	UPTHR[6]	UPTHR[5]	UPTHR[4]	UPTHR[3]	UPTHR[2]	UPTHR[1]	UPTHR[0]	LOWTHR[7]	LOWTHR[6]	LOWTHR[5]	LOWTHR[4]	LOWTHR[3]	LOWTHR[2]	LOWTHR[1]	LOWTHR[0]	REFCLKWND[7]	REFCLKWND[6]	REFCLKWND[5]	REFCLKWND[4]	REFCLKWND[3]	REFCLKWND[2]	REFCLKWND[1]	REFCLKWND[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-70. Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	UPTHR	Upper Threshold Value Register bits These bits define the upper threshold of the counter at the end of each reference clock window. Internally this threshold is multiplied by 4. If the counter value is more than this threshold multiplied by 4, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.
[15:8]	LOWTHR	Lower Threshold Value Register bits These bits define the lower threshold of the counter at the end of each reference clock window. Internally this threshold is multiplied by 4. If the counter value is lower than this threshold multiplied by 4, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Table 2-70. Clock Supervisor Configuration for Main PLL Clock Register (SYSC_CSVMPCFGR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	REFCLKWND	Reference Clock Window Value Register bits These bits define the duration in reference clocks for enabling the measurement of the clock to be monitored. After this time the monitoring clock counter will be compared with its lower and upper limits. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Note: Restrictions for the configuration of the UPTHR, LOWTHR and REFCLKWND bits in the Clock Supervisor Configuration Registers are given in 2.3.7.1 Block diagram.

2.2.9.4 Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR)

Clock Supervisor configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR) configures reference counter window and upper/lower threshold of Main counter for SSCG PLL clock supervision. Reference clock used by SSCG PLL Clock CSV is PLL input clock.

Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR)

Figure 2-70. Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR)

SYSC_CSVSPCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	UPTHR[7]	UPTHR[6]	UPTHR[5]	UPTHR[4]	UPTHR[3]	UPTHR[2]	UPTHR[1]	UPTHR[0]	LOWTHR[7]	LOWTHR[6]	LOWTHR[5]	LOWTHR[4]	LOWTHR[3]	LOWTHR[2]	LOWTHR[1]	LOWTHR[0]	REFCLKWND[7]	REFCLKWND[6]	REFCLKWND[5]	REFCLKWND[4]	REFCLKWND[3]	REFCLKWND[2]	REFCLKWND[1]	REFCLKWND[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-71. Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	UPTHR	Upper Threshold Value Register bits These bits define the upper threshold of the counter at the end of each reference clock window. Internally this threshold is multiplied by 4. If the counter value is more than this threshold multiplied by 4, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.
[15:8]	LOWTHR	Lower Threshold Value Register bits These bits define the lower threshold of the counter at the end of each reference clock window. Internally this threshold is multiplied by 4. If the counter value is lower than this threshold multiplied by 4, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Table 2-71. Clock Supervisor Configuration for SSCG-PLL Clock Register (SYSC_CSVSPCFGR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	REFCLKWND	Reference Clock Window Value Register bits These bits define the duration in reference clocks for enabling the measurement of the clock to be monitored. After this time the monitoring clock counter will be compared with its lower and upper limits. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Note: Restrictions for the configuration of the UPTHR, LOWTHR and REFCLKWND bits in the Clock Supervisor Configuration Registers are given in 2.3.7.1 Block diagram.

2.2.9.5 Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR)

Clock Supervisor configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR) configures reference counter window and upper/lower threshold of Main counter for GFX PLL clock supervision. Reference clock used by Graphics PLL clock CSV is PLL input clock.

Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR)

Figure 2-71. Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR)

SYSC_CSVGPCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	UPTHR[7]	UPTHR[6]	UPTHR[5]	UPTHR[4]	UPTHR[3]	UPTHR[2]	UPTHR[1]	UPTHR[0]	LOWTHR[7]	LOWTHR[6]	LOWTHR[5]	LOWTHR[4]	LOWTHR[3]	LOWTHR[2]	LOWTHR[1]	LOWTHR[0]	REFCLKWND[7]	REFCLKWND[6]	REFCLKWND[5]	REFCLKWND[4]	REFCLKWND[3]	REFCLKWND[2]	REFCLKWND[1]	REFCLKWND[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-72. Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	UPTHR	Upper Threshold Value Register bits These bits define the upper threshold of the counter at the end of each reference clock window. Internally this threshold is multiplied by 4. If the counter value is more than this threshold multiplied by 4, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.
[15:8]	LOWTHR	Lower Threshold Value Register bits These bits define the lower threshold of the counter at the end of each reference clock window. Internally this threshold is multiplied by 4. If the counter value is lower than this threshold multiplied by 4, CSV generates device reset if clock is used. If clock is not used then interrupt will be generated. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Table 2-72. Clock Supervisor Configuration for Graphics PLL Clock Register (SYSC_CSVGPCFGR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	REFCLKWND	Reference Clock Window Value Register bits These bits define the duration in reference clocks for enabling the measurement of the clock to be monitored. After this time the monitoring clock counter will be compared with its lower and upper limits. Reset value of these bits is '0', user needs to configure it before enabling the CSV module.

Note: Restrictions for the configuration of the UPTHR, LOWTHR and REFCLKWND bits in the Clock Supervisor Configuration Registers are given in 2.3.7.1.

2.2.9.6 Clock Supervisor Test Register (SYSC_CSVTESTR)

Clock Supervisor Test Register (SYSC_CSVTESTR) controls the input clock gating for functional testing of all CSV modules.

Clock Supervisor Test Register (SYSC_CSVTESTR)

Figure 2-72. Clock Supervisor Test Register (SYSC_CSVTESTR)

SYSC_CSVTESTR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	GPCLKGATE	SPCLKGATE	MPCLKGATE	SOCLKGATE	MOCLKGATE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-73. Clock Supervisor Test Register (SYSC_CSVTESTR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4]	GPCLKGATE	GFX PLL Clock Gate for CSV Test bit This bit gates the clock for GFX PLL for functional testing. '0': Input clock is not gated and CSV works in normal mode '1': Input clock is gated for CSV testing
[3]	SPCLKGATE	SSCG PLL Clock Gate for CSV Test bit This bit gates the clock for SSCG PLL for functional testing. '0': Input clock is not gated and CSV works in normal mode '1': Input clock is gated for CSV testing
[2]	MPCLKGATE	Main PLL Clock Gate for CSV Test bit This bit gates the clock for main PLL for functional testing. '0': Input clock is not gated and CSV works in normal mode '1': Input clock is gated for CSV testing
[1]	SOCLKGATE	Sub Oscillator Clock Gate for CSV Test bit This bit gates the clock for sub oscillator for functional testing. '0': Input clock is not gated and CSV works in normal mode '1': Input clock is gated for CSV testing
[0]	MOCLKGATE	Main Oscillator Clock Gate for CSV Test bit This bit gates the clock for main oscillator for functional testing. '0': Input clock is not gated and CSV works in normal mode '1': Input clock is gated for CSV testing

2.2.10 Reset control registers

These registers provide trigger bits for software reset, software-triggered hardware reset. It also provides Reset Cause Status Registers for user and BootROM.

2.2.10.1 Reset Control Register (SYSC_RSTCNTR)

Reset Control Register (SYSC_RSTCNTR) provides trigger bits for software reset, software-triggered hard reset and debug/ trace reset.

Reset Control Register (SYSC_RSTCNTR)

Figure 2-73. Reset Control Register (SYSC_RSTCNTR)

SYSC_RSTCNTR																															
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**
DBGR[7]	DBGR[6]	DBGR[5]	DBGR[4]	DBGR[3]	DBGR[2]	DBGR[1]	DBGR[0]	SWHRST[7]	SWHRST[6]	SWHRST[5]	SWHRST[4]	SWHRST[3]	SWHRST[2]	SWHRST[1]	SWHRST[0]	read0	read0	read0	read0	read0	read0	read0	read0	SWRST[7]	SWRST[6]	SWRST[5]	SWRST[4]	SWRST[3]	SWRST[2]	SWRST[1]	SWRST[0]
R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0	R0	R0	R0	R0	R0	R0	R0	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: If all the resets are configured at the same time, software-triggered hard reset takes higher priority and resets the complete device. At same time Reset Cause Register captures all reset causes.

Table 2-74. Reset Control Register (SYSC_RSTCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	DBGR	Debug Reset Register bits User can write in this register to generate debug reset for the device. For generating debug reset, the value written to this register should be 0xDA. This register is cleared automatically after each write. Read to this register always returns 0x00. For effect of debug reset refer to 2.3.3.1 Reset sources.
[23:16]	SWHRST	Software Triggered Hardware Reset Register bits User can write in this register to generate software triggered hardware reset for the device. For generating software triggered hardware reset, the value written to this register should be 0xA5. This register is cleared automatically after each write. Read to this register always returns 0x00. For effect of software triggered hardware reset refer to 2.3.3.1 Reset sources.
[15:8]	read0	-

Table 2-74. Reset Control Register (SYSC_RSTCNTR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	SWRST	Software Reset Register bits User can write in this register to generate software reset for the device. For generating software triggered reset, the value written to this register should be 0x5A. This register is cleared automatically after each write. Read to this register always returns 0x00. For effect of software reset refer to 2.3.3.1 Reset sources.

2.2.10.2 User Reset Cause Register (SYSC_RSTCAUSEUR)

User Reset Cause Register (SYSC_RSTCAUSEUR) provides the cause for the last reset. This register should be cleared by the user before the next reset. If not, the new reset cause will be added over the old one and it will be difficult for the user to find the cause for the latest reset. This register get initialized on PRSTX.

User Reset Cause Register (SYSC_RSTCAUSEUR)

Figure 2-74. User Reset Control Register (SYSC_RSTCAUSEUR)

SYSC_RSTCAUSEUR																															
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**
read0	read0	read0	PD5R	PD4R	PD3R	PD2R	FAKEPDR	read0	read0	read0	read0	read0	read0	read0	SWR	read0	read0	read0	read0	read0	read0	CSVRGP	CSVRSP	CSVRMP	CSVRSC	CSVRMC	WDR	PRFERR	SWHRST	RSTX	PRSTX
R0	R0	R0	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	R0	R0	R0	R0	R0	R0	R0	RW0PS	R0	R0	R0	R0	R0	R0	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS
0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Note: The initial value of this register shown is with respect to PRSTX. Also the PRSTX, RSTX, and PDxR bit will be set, before the device starts booting. For more details refer to 2.3.3.9 Startup after power and external reset.

Table 2-75. User Reset Control Register (SYSC_RSTCAUSEUR) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28]	PD5R	Power Domain 5 Reset Status bit '0': Power domain 5 reset not encountered '1': Power domain 5 reset encountered For cause and effect of power domain 5 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[27]	PD4R	Power Domain 4 Reset Status bit '0': Power domain 4 reset not encountered '1': Power domain 4 reset encountered For cause and effect of power domain 4 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.

Table 2-75. User Reset Control Register (SYSC_RSTCAUSEUR) bits

Bit Position	Bit Field Name	Bit Description
[26]	PD3R	Power Domain 3 Reset Status bit '0': Power domain 3 reset not encountered '1': Power domain 3 reset encountered For cause and effect of power domain 3 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[25]	PD2R	Power Domain 2 Reset Status bit '0': Power domain 2 reset not encountered '1': Power domain 2 reset encountered For cause and effect of power domain 2 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[24]	FAKEPDR	Fake Power Down Reset Status bit This bit is set when SYSC_SPCCFGR:FAKEPWRCNT = '1' and reset is generated for power domain 2 or power domain 3. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[23:17]	read0	-
[16]	SWR	Software Reset Status bit Software reset is asserted by writing 0x5A in SYSC_RSTCNTR:SWRST register. '0': Software reset not encountered '1': Software reset encountered This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[15:10]	read0	-
[9]	CSVRGP	Graphics PLL Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Graphics PLL clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Graphics PLL clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[8]	CSVRSP	SSCG PLL Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered SSCG PLL clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered SSCG PLL clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.

Table 2-75. User Reset Control Register (SYSC_RSTCAUSEUR) bits

Bit Position	Bit Field Name	Bit Description
[7]	CSVRMP	Main PLL Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Main PLL clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Main PLL clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[6]	CSVRSC	Sub Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Sub clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Sub clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[5]	CSVRMC	Main Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Main clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Main clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[4]	WDR	Watchdog Reset Status bit Watchdog overflow/early or wrong trigger detected by device Watchdog timer asserts this reset. '0': No Watchdog reset was asserted '1': Watchdog reset was asserted For cause and effect of watchdog reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[3]	PRFERR	Profile Error Reset Status bit Any invalid RUN profile applied on the device at the time of wakeup sets this bit. This can happen due to bit flip in PSS. '0': No profile error reset was asserted '1': Profile error reset was asserted For cause and effect of profile error refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.

Table 2-75. User Reset Control Register (SYSC_RSTCAUSEUR) bits

Bit Position	Bit Field Name	Bit Description
[2]	SWHRST	Software Triggered Hardware Reset Status bit Software triggered hardware reset is asserted by writing 0xA5 in SYSC_RSTCNTR:SWHRST. '0': No software triggered hardware reset was asserted '1': Software triggered hardware reset was asserted This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[1]	RSTX	External Reset Status bit '0': After power ON, no external reset was asserted '1': After power ON, external reset was asserted This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.
[0]	PRSTX	Power ON Reset Status bit This register bit is set during device power up. This bit should be cleared by writing '0' by user before next reset. Writing '1' to this register bit has no effect.

Note: Reset cause register shows which all resets have been applied, means during hard reset all switchable power-domains are reset and PDxR in the reset-cause register will also be set. So if software wants the real cause of the reset then it has to scan all bits of reset cause register in order of its priority.

2.2.10.3 BootROM Reset Cause Register (SYSC_RSTCAUSEBT)

BootROM Reset Cause Register (SYSC_RSTCAUSEBT) provides the cause for the last reset. This register should be cleared by BootROM routine. At power reset all bits of Reset Cause Register except PRSTX is cleared. This register get initialized on PRSTX.

BootROM Reset Cause Register (SYSC_RSTCAUSEBT)

Figure 2-75. BootROM Reset Cause Register (SYSC_RSTCAUSEBT)

SYSC_RSTCAUSEBT																													
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**
reserved	read0	read0	PD5R	PD4R	PD3R	PD2R	FAKEPDR	read0	read0	read0	read0	read0	read0	read0	SWR	read0	read0	read0	read0	read0	read0	CSVRGP	CSVRSP	CSVRMP	CSVRSC	CSVRMC	WDR	PRFERR	SWHRST	RSTX	PRSTX
-	R0	R0	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	R0	R0	R0	R0	R0	R0	R0	RW0PS	R0	R0	R0	R0	R0	R0	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS
X	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Note: The initial value of this register shown is with respect to PRSTX. The PRSTX, RSTX, and PDxR bit will be set, before the device starts booting. For more details refer to 2.3.3.9 Startup after power and external reset.

Table 2-76. BootROM Reset Cause Register (SYSC_RSTCAUSEBT) bits

Bit Position	Bit Field Name	Bit Description
[31]	reserved	
[30:29]	read0	-
[28]	PD5R	Power Domain 5 Reset Status bit '0': Power domain 5 reset not encountered '1': Power domain 5 reset encountered For cause and effect of power domain 5 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[27]	PD4R	Power Domain 4 Reset Status bit '0': Power domain 4 reset not encountered '1': Power domain 4 reset encountered For cause and effect of power domain 4 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.

Table 2-76. BootROM Reset Cause Register (SYSC_RSTCAUSEBT) bits

Bit Position	Bit Field Name	Bit Description
[26]	PD3R	Power Domain 3 Reset Status bit '0': Power domain 3 reset not encountered '1': Power domain 3 reset encountered For cause and effect of power domain 3 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[25]	PD2R	Power Domain 2 Reset Status bit '0': Power domain 2 reset not encountered '1': Power domain 2 reset encountered For cause and effect of power domain 2 reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[24]	FAKEPDR	Fake Power Down Reset Status bit This bit is set when SYSC_SPCCFGR:FAKEPWRCNT = '1' and reset is generated for power domain 2 or power domain 3. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[23:17]	read0	-
[16]	SWR	Software Reset Status bit Software reset is asserted by writing 0x5A in SYSC_RSTCNTR:SWRST register. '0': Software reset not encountered '1': Software reset encountered This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[15:10]	read0	-
[9]	CSVRGP	Graphics PLL Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Graphics PLL clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Graphics PLL clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.

Table 2-76. BootROM Reset Cause Register (SYSC_RSTCAUSEBT) bits

Bit Position	Bit Field Name	Bit Description
[8]	CSVRSP	SSCG PLL Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered SSCG PLL clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered SSCG PLL clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[7]	CSVRMP	Main PLL Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Main PLL clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Main PLL clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[6]	CSVRSC	Sub Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Sub clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Sub clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[5]	CSVRMC	Main Clock Supervisor Reset Status bit '0': No Clock Supervisor reset encountered Main clock missing or clock unstable which is out of tolerance '1': Clock Supervisor reset encountered Main clock missing or clock unstable which is out of tolerance For cause of CSV reset and there effect refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.

Table 2-76. BootROM Reset Cause Register (SYSC_RSTCAUSEBT) bits

Bit Position	Bit Field Name	Bit Description
[4]	WDR	Watchdog Reset Status bit Watchdog overflow/early or wrong trigger detected by device watch-dog timer asserts this reset. '0': No watchdog reset was asserted '1': Watchdog reset was asserted For cause and effect of watchdog reset refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[3]	PRFERR	Profile Error Reset Status bit Any invalid RUN profile apply on the device at the time of wakeup sets this bit. This can happen due to bit flip in PSS. '0': No profile error reset was asserted '1': Profile error reset was asserted For cause and effect of profile error refer to 2.3.3.1 Reset sources. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[2]	SWHRST	Software Triggered Hardware Reset Status bit Software triggered hardware reset is asserted by writing 0xA5 in SYSC_RSTCNTR:SWHRST. '0': No software triggered hardware reset was asserted '1': Software triggered hardware reset was asserted This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[1]	RSTX	External Reset Status bit '0': After power ON, no external reset was asserted '1': After power ON, external reset was asserted This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.
[0]	PRSTX	Power ON Reset Status bit This register bit is set during device power up. This bit should be cleared by writing '0' by BootROM routine before next reset. Writing '1' to this register bit has no effect.

Note: Reset cause register shows which all resets have been applied, means during hard reset all switchable power-domains are reset and PDxR in the reset-cause register will also be set. So if software wants the real cause of the reset then it has to scan all bits of reset cause register in order of its priority.

2.2.11 Slow RC Source Clock Timer registers

This register group contains Slow RC Source Clock Timer (SCT) Configuration and Status Registers. SYSC_SRCSCTTRG and SYSC_SRCSCTCPR registers are protected by protected sequence only when SCT is used for clock stabilization function. When SCT is used as normal timer, the normal PPU functionality is applied for all SCT registers. These registers get initialized on Hard resets.

To ensure valid stabilization time settings after re-enabling a source clock, regsiters SYSC_SRCSCTTRG, SYSC_SRCCTCNTR and SYSC_SRCSCTCPR are initialized when the Sub Source Clock is not enabled (SYSC_APPCKSRER:SOSCEN is '0'). Thus, writing to these registers cannot happen while the source clock is disabled and will result in an error response.

2.2.11.1 Slow RC SCT Trigger Register (SYSC_SRCSCTTRG)

This register has the configuration capture bit which is used to synchronize and capture the configuration data safely in source timer. Set this bit, after all source clock configuration registers are programmed.

Slow RC SCT Trigger Register (SYSC_SRCSCTTRG)

Figure 2-76. Slow RC SCT Trigger Register (SYSC_SRCSCTTRG)

SYSC_SRCSCTTRG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CGCPT
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-77. Slow RC SCT Trigger Register (SYSC_SRCSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-

Table 2-77. Slow RC SCT Trigger Register (SYSC_SRCSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[0]	CGCPT	Timer Configuration Capture Enable This bit has to be set for any updated configuration control data to be captured within Slow RC Source Clock Timer module. '0': No operation '1': Triggers Source Clock Timer to capture new timer configuration Writing '0' in this bit will have no effect. Once Source Clock Timer is triggered for new configuration capture, SYSC_SRCSCTS-TATR:BUSY bit is set. This bit is cleared by hardware once configuration is captured by SCT. This bit has to be programmed with protected sequence, during Slow RC clock stabilization where R0WPS1 attribute is active. After Slow RC clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. Reading this bit always returns '0'.

2.2.11.2 Slow RC SCT Control Register (SYSC_SRCSCTCNTR)

This register controls timer mode of operation. This register has debug bit, which when it is set, halts the Source Clock Timer at break points. This register is locked during Slow RC clock stabilization, which means software cannot modify its default value during this time. For SCT configuration sequence refer to 2.4.5 Notes on using Source Clock Timer.

Slow RC SCT Control Register (SYSC_SRCSCTCNTR)

Figure 2-77. Slow RC SCT Control Register (SYSC_SRCSCTCNTR)

SYSC_SRCSCTCNTR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGEN	MODE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-78. Slow RC SCT Control Register (SYSC_SRCSCTCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	DBGEN	Debug Enable bit This bit is used to select the counter behavior during break point. '0': Counter can run during a break point '1': Counter is stopped during a break point During Slow RC clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After Slow RC clock stabilization, RpWp attribute is active.
[0]	MODE	Mode Control This bit is used to control the operating mode of Slow RC Source Clock Timer. '0': Single shot mode '1': Continuous mode During Slow RC clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After Slow RC clock stabilization, RpWp attribute is active.

2.2.11.3 Slow RC SCT Compare Prescaler Register (SYSC_SRCSCTCPR)

This register is used to configure timer prescalar and the timer duration values.

Slow RC SCT Compare Prescaler Register (SYSC_SRCSCTCPR)

Figure 2-78. Slow RC SCT Compare Prescale Register (SYSC_SRCSCTCPR)

SYSC_SRCSCTCPR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PSCL[3]	PSCL[2]	PSCL[1]	PSCL[0]	CMPR[15]	CMPR[14]	CMPR[13]	CMPR[12]	CMPR[11]	CMPR[10]	CMPR[9]	CMPR[8]	CMPR[7]	CMPR[6]	CMPR[5]	CMPR[4]	CMPR[3]	CMPR[2]	CMPR[1]	CMPR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 2-79. Slow RC SCT Compare Prescale Register (SYSC_SRCSCTCPR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:20]	read0	-
[19:16]	PSCL	Prescale Control Value This bit is used to select clock prescale. '0000': No clock division '0001': Clock division by 2 '0010': Clock division by 4 ... '1000': Clock division by 256 '1001': Clock division by 512 ... '1111': Clock division by 32768 These bits have to be programmed with protected sequence, during SRC clock stabilization where RWPS attribute is active. After Slow RC clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.
[15:0]	CMPR	Timer Compare Value These bits store 16-bit data, which is compared with timer value. Comparison is true if the timer reaches value greater than or equal to the compare value. These bits have to be programmed with protected sequence, during Slow RC clock stabilization where RWPS attribute is active. After Slow RC clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.

Notes:

1. In single shot mode there can be inaccuracy of less than half of the pre-scalar value in interval between trigger and interrupt.
2. To avoid the inaccuracy of first interval at reconfiguration, we need to clear the counter by setting a dummy sinle-shot mode with ‘1’ compare value and disabled-interrupt.
3. Do not trigger timer with compare value as 0, otherwise timer runs forever.

2.2.11.4 Slow RC SCT Status Register (SYSC_SRCSCTSTATR)

This register shows the current timer status.

Slow RC SCT Status Register (SYSC_SRCSCTSTATR)

Figure 2-79. Slow RC SCT Status Register (SYSC_SRCSCTSTATR)

SYSC_SRCSCTSTATR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	BUSY	TRSTS	INTF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-80. Slow RC SCT Status Register (SYSC_SRCSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:3]	read0	-
[2]	BUSY	Configuration Update Status This read-only bit is used for configuration update status. '0': The configuration data has been synchronized and updated within source timer module '1': The configuration data update within source timer module is in progress User should not update configuration registers if this bit reads '1'. If done SCT generates PPU error. Above rule does not apply to the SYSC_SRCSCTINTER and SYSC_SRCSCTICLR registers. This bit goes high as soon as SYSC_SRCSCTTRG:CGCPT bit is set and will remain high till configuration data is synchronized and updated within source timer module. Writing this bit has no effect. During Slow RC clock stabilization where R attribute is active, after Slow RC clock stabilization Rp attribute is active.
[1]	TRSTS	Source Clock Timer Status This read-only bit is used for source timer status. '0': Timer not active, meaning the counter has stopped incrementing '1': Timer active, meaning the counter is incrementing Writing this bit has no effect. During Slow RC clock stabilization where R attribute is active, after Slow RC clock stabilization Rp attribute is active.

Table 2-80. Slow RC SCT Status Register (SYSC_SRCSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[0]	INTF	Interrupt Status Flag This interrupt flag is set when the timer reaches value greater than or equal to the compare value programmed in SYSC_SRCSCTCPR:CMPR. This bit will be set only if clock stabilization time has elapsed and the Source Clock Timer is used as a normal timer. Timer interrupt can be cleared by writing interrupt clear bit SYSC_SRCSCTICLR:INTC bit to '1'. During Slow RC clock stabilization where R attribute is active, after Slow RC clock stabilization Rp attribute is active.

2.2.11.5 Slow RC SCT Interrupt Enable Register (SYSC_SRCSCTINTER)

This register enables Slow RC Source Clock Timer interrupt.

Slow RC SCT Interrupt Enable Register (SYSC_SRCSCTINTER)

Figure 2-80. Slow RC SCT Interrupt Enable Register (SYSC_SRCSCTINTER)

SYSC_SRCSCTINTER																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-81. Slow RC SCT Interrupt Enable Register (SYSC_SRCSCTINTER) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTE	Interrupt Enable bit This bit is used to enable the source timer interrupt request signal. '0': Interrupt is not generated for SYSC_SRCSCTSTATR:INTF = '1' '1': Interrupt is generated if SYSC_SRCSCTSTATR:INTF = '1' This bit has to be programmed with protected sequence, during SRC clock stabilization where RWPS attribute is active. After Slow RC clock stabilization, this bit can be programmed normally (without protected sequence) where RpWp attribute is active.

2.2.11.6 Slow RC SCT Interrupt Clear Register (SYSC_SRCSCTICLR)

This register clears the Slow RC Source Clock Timer interrupt.

Slow RC SCT Interrupt Clear Register (SYSC_SRCSCTICLR)

Figure 2-81. Slow RC SCT Interrupt Clear Register (SYSC_SRCSCTICLR)

SYSC_SRCSCTICLR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTC
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-82. Slow RC SCT Interrupt Clear Register (SYSC_SRCSCTICLR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTC	Interrupt Request Clear bit This bit is used to clear the interrupt flag. Writing '1' in this register bit clears SYSC_SRCSCTSTATR:INTF. Writing '0' has no effect. This bit has to be programmed with protected sequence, during SRC clock stabilization where R0WPS1 attribute is active. After Slow RC clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. This bit is cleared by hardware.

2.2.12 RC Source Clock Timer registers

This register group contains RC Source Clock Timer Configuration and Status Registers. SYSC_RCSCTTRG and SYSC_RCSCTCPR registers are protected by protected sequence only when SCT is used for clock stabilization function. When SCT is used as normal timer, the normal PPU functionality is applied for all SCT registers. These registers get initialized on hard resets.

To ensure valid stabilization time settings after re-enabling a source clock, regsiters SYSC_RCSCTTRG, SYSC_RCCTCNTR and SYSC_RCSCTCPR are initialized when the Sub Source Clock is not enabled (SYSC_APPCKSRER:SOSCEN is '0'). Thus, writing to these registers cannot happen while the source clock is disabled and will result in an error response.

2.2.12.1 RC SCT Trigger Register (SYSC_RCSCTTRG)

This register has the configuration capture bit which is used to synchronize and capture the configuration data safely in SOURCE_TIMER. Set this bit, after all RC Source Clock Configuration Registers are programmed.

RC SCT Trigger Register (SYSC_RCSCTTRG)

Figure 2-82. RC SCT Trigger Register (SYSC_RCSCTTRG)

SYSC_RCSCTTRG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CGCPT
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-83. RC SCT Trigger Register (SYSC_RCSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-

Table 2-83. RC SCT Trigger Register (SYSC_RCSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[0]	CGCPT	Timer Configuration Capture Enable This bit has to be set for any updated configuration control data to be captured within RC Source Clock Timer module. '0': No operation '1': Triggers Source Clock Timer to capture new timer configuration Writing '0' in this bit will have no effect. Once Source Clock Timer is triggered for new configuration capture, SYSC_RCSCTSTATR:BUSY bit is set. This bit is cleared by hardware once configuration is captured by SCT. This bit has to be programmed with protected sequence, during RC clock stabilization where R0WPS1 attribute is active. After RC clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. Reading this bit always returns '0'.

2.2.12.2 RC SCT Control Register (SYSC_RCSCTCNTR)

This register controls timer mode of operation. This register has debug bit, which when it is set, halts the Source Clock Timer at break points. This register is locked during RC clock stabilization, which means software cannot modify its default value during this time. For SCT configuration sequence refer to 2.4.5 Notes on using Source Clock Timer.

RC SCT Control Register (SYSC_RCSCTCNTR)

Figure 2-83. RC SCT Control Register (SYSC_RCSCTCNTR)

SYSC_RCSCTCNTR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGEN	MODE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-84. RC SCT Control Register (SYSC_RCSCTCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	DBGEN	Debug Enable bit This bit is used to select the counter behavior during break point. '0': Counter can run during a break point '1': Counter is stopped during a break point During RC clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After RC clock stabilization, RpWp attribute is active.
[0]	MODE	Mode Control This bit is used to control the operating mode of RC Source Clock Timer. '0': Single shot mode '1': Continuous mode During RC clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After RC clock stabilization, RpWp attribute is active.

2.2.12.3 RC SCT Compare Prescaler Register (SYSC_RCSCTCPR)

This register is used to configure the timer prescaler and the timer duration values.

RC SCT Compare Prescaler Register (SYSC_RCSCTCPR)

Figure 2-84. RC SCT Compare Prescale Register (SYSC_RCSCTCPR)

SYSC_RCSCTCPR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PSCL[3]	PSCL[2]	PSCL[1]	PSCL[0]	CMPR[15]	CMPR[14]	CMPR[13]	CMPR[12]	CMPR[11]	CMPR[10]	CMPR[9]	CMPR[8]	CMPR[7]	CMPR[6]	CMPR[5]	CMPR[4]	CMPR[3]	CMPR[2]	CMPR[1]	CMPR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0

Table 2-85. RC SCT Compare Prescale Register (SYSC_RCSCTCPR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:20]	read0	-
[19:16]	PSCL	Prescale Control Value This bit is used to select clock prescale. '0000': No clock division '0001': Clock division by 2 '0010': Clock division by 4 ... '1000': Clock division by 256 '1001': Clock division by 512 ... '1111': Clock division by 32768 These bits have to be programmed with protected sequence, during RC clock stabilization where RWPS attribute is active. After RC clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.
[15:0]	CMPR	Timer Compare Value These bits store 16-bit data, which is compared with timer value. Comparison is true if the timer reaches value greater than or equal to the compare value. These bits have to be programmed with protected sequence, during RC clock stabilization where RWPS attribute is active. After RC clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.

Notes:

1. In single shot mode there can be inaccuracy of less than half of the pre-scalar value in interval between trigger and interrupt.
2. To avoid the inaccuracy of first interval at reconfiguration, we need to clear the counter by setting a dummy sinle-shot mode with ‘1’ compare value and disabled-interrupt.
3. Do not trigger timer with compare value as 0, otherwise timer runs forever.

2.2.12.4 RC SCT Status Register (SYSC_RCSCTSTATR)

This register shows the current timer status.

RC SCT Status Register (SYSC_RCSCTSTATR)

Figure 2-85. RC SCT Status Register (SYSC_RCSCTSTATR)

SYSC_RCSCTSTATR																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	BUSY	TRSTS	INTF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-86. RC SCT Status Register (SYSC_RCSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:3]	read0	-
[2]	BUSY	Configuration Update Status This read-only bit is used for configuration update status. '0': The configuration data has been synchronized and updated within source timer module '1': The configuration data update within source timer module is in progress User should not update configuration registers if this bit reads '1'. If done SCT generates PPU error. Above rule does not apply to the SYSC_RCSCTINTER and SYSC_RCSCTICLR registers. This bit goes high as soon as SYSC_RCSCTTRG:CGCPT bit is set and will remain high till configuration data is synchronized and updated within source timer module. Writing this bit has no effect. During RC clock stabilization where R attribute is active, after RC clock stabilization Rp attribute is active.
[1]	TRSTS	Source Clock Timer Status This read-only bit is used for source timer status. '0': Timer not active, meaning the counter has stopped incrementing '1': Timer active, meaning the counter is incrementing Writing this bit has no effect. During RC clock stabilization where R attribute is active, after RC clock stabilization Rp attribute is active.

Table 2-86. RC SCT Status Register (SYSC_RCSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[0]	INTF	Interrupt Status Flag This interrupt flag is set when the timer reaches value greater than or equal to the compare value programmed in SYSC_RCSCTCPR:CMPR. This bit will be set only if clock stabilization time has elapsed and the Source Clock Timer is used as a normal timer. Timer interrupt can be cleared by writing interrupt clear bit SYSC_RCSCTICLR:INTC bit to '1'. During RC clock stabilization where R attribute is active, after RC clock stabilization Rp attribute is active.

2.2.12.5 RC SCT Interrupt Enable Register (SYSC_RCSCTINTER)

This register enables RC Source Clock Timer interrupt.

RC SCT Interrupt Enable Register (SYSC_RCSCTINTER)

Figure 2-86. RC SCT Interrupt Enable Register (SYSC_RCSCTINTER)

SYSC_RCSCTINTER																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-87. RC SCT Interrupt Enable Register (SYSC_RCSCTINTER) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTE	Interrupt Enable bit This bit is used to enable the source timer interrupt request signal. '0': Interrupt is not generated if SYSC_RCSCTSTATR:INTF = '1' '1': Interrupt is generated for SYSC_RCSCTSTATR:INTF = '1' This bit has to be programmed with protected sequence, during RC clock stabilization where RWPS attribute is active. After RC clock stabilization, this bit can be programmed normally (without protected sequence) where RpWp attribute is active.

2.2.12.6 RC SCT Interrupt Clear Register (SYSC_RCSCTICLR)

This register clears the RC Source Clock Timer interrupt.

RC SCT Interrupt Clear Register (SYSC_RCSCTICLR)

Figure 2-87. RC SCT Interrupt Clear Register (SYSC_RCSCTICLR)

SYSC_RCSCTICLR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTC
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-88. RC SCT Interrupt Clear Register (SYSC_RCSCTICLR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTC	Interrupt Request Clear bit This bit is used to clear the interrupt flag. Writing '1' in this register bit clears SYSC_RCSCTSTATR:INTF. Writing '0' has no effect. This bit has to be programmed with protected sequence, during RC clock stabilization where R0WPS1 attribute is active. After RC clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. This bit is cleared by hardware.

2.2.13 Main Source Clock Timer Registers

This register group contains Main Source Clock Timer Configuration and Status Registers. SYSC_MAINSCTTRG and SYSC_MAINSCTCPR registers are protected by protected sequence only when SCT is used for clock stabilization function. When SCT is used as normal timer, the normal PPU functionality is applied for all SCT registers. These registers are initialized on Hard resets.

To ensure valid stabilization time settings after re-enabling a source clock, regsiters SYSC_MAINSCTTRG, SYSC_MAINCTCNTR and SYSC_MAINSCTCPR are initialized when the Sub Source Clock is not enabled (SYSC_APPCKSRER:SOSCEN is '0'). Thus, writing to these registers cannot happen while the source clock is disabled and will result in an error response.

2.2.13.1 Main SCT Trigger Register (SYSC_MAINSCTTRG)

This register has the configuration capture bit which is used to synchronize and capture the configuration data safely in SOURCE_TIMER. Set this bit, after all Main Source Clock Configuration registers are programmed.

Main SCT Trigger Register (SYSC_MAINSCTTRG)

Figure 2-88. Main SCT Trigger Register (SYSC_MAINSCTTRG)

SYSC_MAINSCTTRG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CGCPT
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-89. Main SCT Trigger Register (SYSC_MAINSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-

Table 2-89. Main SCT Trigger Register (SYSC_MAINSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[0]	CGCPT	Timer Configuration Capture Enable This bit has to be set for any updated configuration control data to be captured within Main Source Clock Timer module. '0': No operation '1': Triggers Source Clock Timer to capture new timer configuration Writing '0' in this bit will have no effect. Once Source Clock Timer is triggered for new configuration capture, SYSC_MAINSCTS-TATR:BUSY bit is set. This bit is cleared by hardware once configuration is captured by SCT. This bit has to be programmed with protected sequence, during Main clock stabilization where R0WPS1 attribute is active. After Main clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. Reading this bit always returns '0'.

2.2.13.2 Main SCT Control Register (SYSC_MAINSCTCNTR)

This register controls timer mode of operation. This register has debug bit, which when set, halts the Source Clock Timer at break points. This register is locked during Main clock stabilization, which means software cannot modify its default value during this time. For SCT configuration sequence refer to 2.4.5 Notes on using Source Clock Timer.

Main SCT Control Register (SYSC_MAINSCTCNTR)

Figure 2-89. Main SCT Control Register (SYSC_MAINSCTCNTR)

SYSC_MAINSCTCNTR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGEN	MODE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-90. Main SCT Control Register (SYSC_MAINSCTCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	DBGEN	Debug Enable bit This bit is used to select the counter behavior during break point. '0': Counter can run during a break point '1': Counter is stopped during a break point During Main clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After Main clock stabilization, RpWp attribute is active.
[0]	MODE	Mode Control This bit is used to control the operating mode of Main Source Clock Timer. '0': Single shot mode '1': Continuous mode During Main clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After Main clock stabilization, RpWp attribute is active.

2.2.13.3 Main SCT Compare Prescaler Register (SYSC_MAINSCTCPR)

This register is used to configure timer prescaler and the timer duration values.

Main SCT Compare Prescaler Register (SYSC_MAINSCTCPR)

Figure 2-90. Main SCT Compare Prescale Register (SYSC_MAINSCTCPR)

SYSC_MAINSCTCPR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PSCL[3]	PSCL[2]	PSCL[1]	PSCL[0]	CMPR[15]	CMPR[14]	CMPR[13]	CMPR[12]	CMPR[11]	CMPR[10]	CMPR[9]	CMPR[8]	CMPR[7]	CMPR[6]	CMPR[5]	CMPR[4]	CMPR[3]	CMPR[2]	CMPR[1]	CMPR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-91. Main SCT Compare Prescale Register (SYSC_MAINSCTCPR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:20]	read0	-
[19:16]	PSCL	Prescale Control Value This bit is used to select clock prescale. '0000': No clock division '0001': Clock division by 2 '0010': Clock division by 4 ... '1000': Clock division by 256 '1001': Clock division by 512 ... '1111': Clock division by 32768 These bits have to be programmed with protected sequence, during Main clock stabilization where RWPS attribute is active. After Main clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active. To reduce the Default Main Osc Stabilization time, the prescaler value for devices other than ATLAS and CALYPSO devices is now reduced to 0x5.

Table 2-91. Main SCT Compare Prescale Register (SYSC_MAINSCTCPR) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	CMPR	Timer Compare Value These bits store 16-bit data, which is compared with timer value. Comparison is true if the timer reaches value greater than or equal to the compare value. These bits have to be programmed with protected sequence, during Main clock stabilization where RWPS attribute is active. After Main clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.

Notes:

1. In single shot mode there can be inaccuracy of less than half of the pre-scalar value in interval between trigger and interrupt.
2. To avoid the inaccuracy of first interval at reconfiguration, we need to clear the counter by setting a dummy sinle-shot mode with ‘1’ compare value and disabled-interrupt.
3. Do not trigger timer with compare value as 0, otherwise timer runs forever.

2.2.13.4 Main SCT Status Register (SYSC_MAINSCTSTATR)

This register shows the current timer status.

Main SCT Status Register (SYSC_MAINSCTSTATR)

Figure 2-91. Main SCT Status Register (SYSC_MAINSCTSTATR)

SYSC_MAINSCTSTATR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	BUSY	TRSTS	INTF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-92. Main SCT Status Register (SYSC_MAINSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:3]	read0	-
[2]	BUSY	Configuration Update Status This read-only bit is used for configuration update status. '0': The configuration data has been synchronized and updated within source timer module '1': The configuration data update within source timer module is in progress User should not update configuration registers if this bit reads '1'. If done SCT generates PPU error. Above rule does not apply to the SYSC_MAINSCTINTER and SYSC_MAINSCTICLR registers. This bit goes high as soon as SYSC_MAINSCTTRG:CGCPT bit is set and will remain high till configuration data is synchronized and updated within source timer module. Writing this bit has no effect. During Main clock stabilization where R attribute is active, after Main clock stabilization Rp attribute is active.
[1]	TRSTS	Source Clock Timer Status This read-only bit is used for source timer status. '0': Timer not active, meaning the counter has stopped incrementing '1': Timer active, meaning the counter is incrementing Writing this bit has no effect. During Main clock stabilization where R attribute is active, after Main clock stabilization Rp attribute is active.

Table 2-92. Main SCT Status Register (SYSC_MAINSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[0]	INTF	Interrupt Status Flag This interrupt flag is set when the timer reaches value greater than or equal to the compare value programmed in SYSC_MAIN-SCTCPR:CMPR. This bit will be set only if clock stabilization time has elapsed and the Source Clock Timer is used as a normal timer. Timer interrupt can be cleared by writing interrupt clear bit SYSC_-MAINSCTICLR:INTC to '1'. During Main clock stabilization where R attribute is active, after Main clock stabilization Rp attribute is active.

2.2.13.5 Main SCT Interrupt Enable Register (SYSC_MAINSCTINTER)

This register enables Main Source Clock Timer interrupt.

Main SCT Interrupt Enable Register (SYSC_MAINSCTINTER)

Figure 2-92. Main SCT Interrupt Enable Register (SYSC_MAINSCTINTER)

SYSC_MAINSCTINTER																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-93. Main SCT Interrupt Enable Register (SYSC_MAINSCTINTER) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTE	Interrupt Enable bit This bit is used to enable the source timer interrupt request signal. '0': Interrupt is not generated for SYSC_MAINSCTSTATR:INTF = '1' '1': Interrupt is generated if SYSC_MAINSCTSTATR:INTF = '1' This bit has to be programmed with protected sequence, during Main clock stabilization where RWPS attribute is active. After Main clock stabilization, this bit can be programmed normally (without protected sequence) where RpWp attribute is active.

2.2.13.6 Main SCT Interrupt Clear Register (SYSC_MAINSCTICLR)

This register clears the Main Source Clock Timer interrupt.

Main SCT Interrupt Clear Register (SYSC_MAINSCTICLR)

Figure 2-93. Main SCT Interrupt Clear Register (SYSC_MAINSCTICLR)

SYSC_MAINSCTICLR																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTC
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-94. Main SCT Interrupt Clear Register (SYSC_MAINSCTICLR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTC	Interrupt Request Clear bit This bit is used to clear the interrupt flag. Writing '1' in this bit clears SYSC_MAINSCTSTATR:INTF. Writing '0' has no effect. This bit has to be programmed with protected sequence, during Main clock stabilization where R0WPS1 attribute is active. After Main clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. This bit is cleared by hardware.

2.2.14 Sub Source Clock Timer Registers

This register group contains Sub Source Clock Timer Configuration and Status Registers. SYSC_SUBSCTTRG and SYSC_-SUBSCTCPR registers are protected by protected sequence only when SCT is used for clock stabilization function. When SCT is used as normal timer, the normal PPU functionality is applied for all SCT registers. These registers gets initialized on hard resets.

To ensure valid stabilization time settings after re-enabling a source clock, regsiters SYSC_SUBSCTTRG, SYSC_-SUBCTCNTR and SYSC_SUBSCTCPR are initialized when the Sub Source Clock is not enabled (SYSC_APPCKS-RER:SOSCEN is '0'). Thus, writing to these registers cannot happen while the source clock is disabled and will result in an error response.

An error response is generated when there is a write access to an SCT register which is disabled because the corresponding clock source is disabled.

2.2.14.1 Sub SCT Trigger Register (SYSC_SUBSCTTRG)

This register has the configuration capture bit which is used to synchronize and capture the configuration data safely in SOURCE_TIMER. Set this bit, after all Source Clock Configuration registers are programmed.

Sub SCT Trigger Register (SYSC_SUBSCTTRG)

Figure 2-94. Sub SCT Trigger Register (SYSC_SUBSCTTRG)

SYSC_SUBSCTTRG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CGCPT
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-95. Sub SCT Trigger Register (SYSC_SUBSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-

Table 2-95. Sub SCT Trigger Register (SYSC_SUBSCTTRG) bits

Bit Position	Bit Field Name	Bit Description
[0]	CGCPT	Timer Configuration Capture Enable This bit has to be set for any updated configuration control data to be captured within Sub Source Clock Timer module. '0': No operation '1': Triggers Source Clock Timer to capture new timer configuration Writing '0' in this bit will have no effect. Once Source Clock Timer is triggered for new configuration capture, SYSC_SUBSCTS-TATR:BUSY bit is set. This bit is cleared by hardware once configuration is captured by SCT. This bit has to be programmed with protected sequence, during Sub clock stabilization where R0WPS1 attribute is active. After Sub clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. Reading this bit always returns '0'.

2.2.14.2 Sub SCT Control Register (SYSC_SUBSCTCNTR)

This register controls timer mode of operation. This register has debug bit, which when set, halts the Source Clock Timer at break points. This register is locked during Sub clock stabilization, which means software cannot modify its default value during this time. Refer to 2.4.5 Notes on using Source Clock Timer for SCT configuration sequence.

Sub SCT Control Register (SYSC_SUBSCTCNTR)

Figure 2-95. Sub SCT Control Register (SYSC_SUBSCTCNTR)

SYSC_SUBSCTCNTR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGEN	MODE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-96. Sub SCT Control Register (SYSC_SUBSCTCNTR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	DBGEN	Debug Enable bit This bit is used to select the counter behavior during break point. '0': Counter can run during a break point '1': Counter is stopped during a break point During Sub clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After Sub clock stabilization, RpWp attribute is active.
[0]	MODE	Mode Control This bit is used to control the operating mode of Sub Source Clock Timer. '0': Single shot mode '1': Continuous mode During Sub clock stabilization R attribute is active and write to this register during stabilization generates PPU error. After Sub clock stabilization, RpWp attribute is active.

2.2.14.3 Sub SCT Compare Prescaler Register (SYSC_SUBSCTCPR)

This register is used to configure timer prescaler and the timer duration values.

Sub SCT Compare Prescaler Register (SYSC_SUBSCTCPR)

Figure 2-96. Sub SCT Compare Prescale Register (SYSC_SUBSCTCPR)

SYSC_SUBSCTCPR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PSCL[3]	PSCL[2]	PSCL[1]	PSCL[0]	CMPR[15]	CMPR[14]	CMPR[13]	CMPR[12]	CMPR[11]	CMPR[10]	CMPR[9]	CMPR[8]	CMPR[7]	CMPR[6]	CMPR[5]	CMPR[4]	CMPR[3]	CMPR[2]	CMPR[1]	CMPR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Table 2-97. Sub SCT Compare Prescale Register (SYSC_SUBSCTCPR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:20]	read0	-
[19:16]	PSCL	Prescale Control Value This bit is used to select clock prescale. '0000': No clock division '0001': Clock division by 2 '0010': Clock division by 4 ... '1000': Clock division by 256 '1001': Clock division by 512 ... '1111': Clock division by 32768 These bits have to be programmed with protected sequence, during Sub clock stabilization where RWPS attribute is active. After Sub clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.
[15:0]	CMPR	Timer Compare Value These bits store 16-bit data, which is compared with timer value. Comparison is true if the timer reaches value greater than or equal to the compare value. These bits have to be programmed with protected sequence, during Sub clock stabilization where RWPS attribute is active. After Sub clock stabilization, these bits can be programmed normally (without protected sequence) where RpWp attribute is active.

Notes:

1. In single shot mode there can be inaccuracy of less than half of the pre-scalar value in interval between trigger and interrupt.
2. To avoid the inaccuracy of first interval at reconfiguration, we need to clear the counter by setting a dummy sinle-shot mode with ‘1’ compare value and disabled-interrupt.
3. Do not trigger timer with compare value as 0, otherwise timer runs forever.

2.2.14.4 Sub SCT Status Register (SYSC_SUBSCTSTATR)

This register shows the current timer status.

Sub SCT Status Register (SYSC_SUBSCTSTATR)

Figure 2-97. Sub SCT Status Register (SYSC_SUBSCTSTATR)

SYSC_SUBSCTSTATR																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	BUSY	TRSTS	INTF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-98. Sub SCT Status Register (SYSC_SUBSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:3]	read0	-
[2]	BUSY	Configuration Update Status This read-only bit is used for configuration update status. '0': The configuration data has been synchronized and updated within source timer module '1': The configuration data update within source timer module is in progress User should not update configuration registers if this bit reads '1'. If done SCT generates PPU error. Above rule does not apply to the SYSC_SUBSCTINTER and SYSC_-SUBSCTICLR registers. This bit goes high as soon as SYSC_SUBSCTTRG:CGCPT bit is set and will remain high till configuration data is synchronized and updated within source timer module. Writing this bit has no effect. During Sub clock stabilization where R attribute is active, after Sub clock stabilization Rp attribute is active.
[1]	TRSTS	Source Clock Timer Status This read-only bit is used for source timer status. '0': Timer not active, meaning the counter has stopped incrementing '1': Timer active, meaning the counter is incrementing Writing this bit has no effect. During Sub clock stabilization where R attribute is active, after Sub clock stabilization Rp attribute is active.

Table 2-98. Sub SCT Status Register (SYSC_SUBSCTSTATR) bits

Bit Position	Bit Field Name	Bit Description
[0]	INTF	Interrupt Status Flag This interrupt flag is set when the timer reaches value greater than or equal to the compare value programmed in SYSC_SUB-SCTCPR:CMPR. This bit will be set only if clock stabilization time has elapsed and the Source Clock Timer is used as a normal timer. Timer interrupt can be cleared by writing interrupt clear bit SYSC_-SUBSCTICLR:INTC bit to '1'. During Sub clock stabilization where R attribute is active, after Sub clock stabilization Rp attribute is active.

2.2.14.5 Sub SCT Interrupt Enable Register (SYSC_SUBSCTINTER)

This register enables Sub Source Clock Timer interrupt.

Sub SCT Interrupt Enable Register (SYSC_SUBSCTINTER)

Figure 2-98. Sub SCT Interrupt Enable Register (SYSC_SUBSCTINTER)

SYSC_SUBSCTINTER																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWPpS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-99. Sub SCT Interrupt Enable Register (SYSC_SUBSCTINTER) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTE	Interrupt Enable bit This bit is used to enable the source timer interrupt request signal. '0': Interrupt is not generated for SYSC_SUBSCTSTATR:INTF = '1' '1': Interrupt is generated if SYSC_SUBSCTSTATR:INTF = '1' This bit has to be programmed with protected sequence, during Sub clock stabilization where RWPS attribute is active. After Sub clock stabilization, this bit can be programmed normally (without protected sequence) where RpWp attribute is active.

2.2.14.6 Sub SCT Interrupt Clear Register (SYSC_SUBSCTICLR)

This register clears the Sub Source Clock Timer interrupt.

Sub SCT Interrupt Clear Register (SYSC_SUBSCTICLR)

Figure 2-99. Sub SCT Interrupt Clear Register (SYSC_SUBSCTICLR)

SYSC_SUBSCTICLR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTC
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0WPpS1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-100. Sub SCT Interrupt Clear Register (SYSC_SUBSCTICLR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	INTC	Interrupt Request Clear bit This bit is used to clear the interrupt flag. Writing '1' in this register bit clears SYSC_SUBSCTSTATR:INTF. Writing '0' has no effect. This bit has to be programmed with protected sequence, during Sub clock stabilization where R0WPS1 attribute is active. After Sub clock stabilization, this bit can be programmed normally (without protected sequence) where Rp0Wp1 attribute is active. This bit is cleared by hardware.

2.2.15 Clock output function configuration registers

The Clock output function configuration registers controls the clock select and prescaler value for clock output function.

2.2.15.1 Clock Output Function Configuration Register (SYSC_CKOTCFGR)

Clock Output Function Configuration Register (SYSC_CKOTCFGR) configures the clock selection and prescaler value for clock output function.

Clock Output Function Configuration Register (SYSC_CKOTCFGR)

Figure 2-100. Clock Output Function Configuration Register (SYSC_CKOTCFGR)

SYSC_CKOTCFGR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CKOUTDIV[2]	CKOUTDIV[1]	CKOUTDIV[0]	read0	read0	read0	read0	read0	CKSEL[2]	CKSEL[1]	CKSEL[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 2-101. Clock Output Function Configuration Register (SYSC_CKOTCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10:8]	CKOUTDIV	Clock Division bits These bits configure the clock output divider according to the following table. '000': Output clock is not divided '001': Output clock is divided by 2 '010': Output clock is divided by 4 '011': Output clock is divided by 8 '100': Output clock is divided by 16 '101': Output clock is divided by 32 '110': Output clock is divided by 64 ' 111': Output clock is divided by 128 Any reset initializes these bits to '000' (output clock is not divided).
[7:3]	read0	-

Table 2-101. Clock Output Function Configuration Register (SYSC_CKOTCFGR) bits

Bit Position	Bit Field Name	Bit Description
[2:0]	CKSEL	Clock Select bits These bits select which clock is output to the CKOT pins according to the following table. '000': Slow RC clock is selected '001': RC clock is selected '010': Sub oscillator clock is selected '011': Main oscillator clock is selected '100': Main PLL clock is selected '101': SSCG PLL clock is selected '110': GFX PLL clock is selected '111': Clock is tied to low Any reset initializes these bits to '111' (no clock is selected).

2.2.16 Special configuration registers

The Special configuration register has few special settings (fast clock input, fake power down mode, PLL stabilization time, RC clock frequency configuration, test bits, etc.).

2.2.16.1 Special Configuration Register (SYSC_SPCCFGR)

Special Configuration Register (SYSC_SPCCFGR) has few special settings (fast clock input, fake power down mode, PLL stabilization time, etc.).

Special Configuration Register (SYSC_SPCCFGR)

Figure 2-101. Special Configuration Register (SYSC_SPCCFGR)

SYSC_SPCCFGR																													
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**
read0	read0	read0	read0	read0	read0	read0	HOLDIO	PSSPADCTRL	read0	GPIL[1]	GPIL[0]	read0	read0	reserved	FCIMEN	read0	read0	read0	PLLINSEL	read0	GFXSTABS	SSCGSTABS	PLLSTABS	read0	read0	read0	read0	read0	read0	FAKEPWRCNT	FASTON
R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	RWPS	RWPS	R0	R0	-	RWPS	R0	R0	R0	RWPS	R0	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-102. Special Configuration Register (SYSC_SPCCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	HOLDIO	Hold IO Control bit This bit controls the isolation latch enable. '0': Isolation latch is transparent '1': Isolation latch holds the previous value This bit is needed for the purpose of retaining IO data at power save state, if PD2 is switched OFF. This bit needs to be set before entering into PSS state with PD2 switched OFF. This bit when set also enables SYSC_SPCCFGR:PSSPADCTRL and SYSC_SPCCFGR:GPIL bit functionality. For more details refer to Section 'IO data retention at power off on this bit programming sequence'.

Table 2-102. Special Configuration Register (SYSC_SPCCFGR) bits

Bit Position	Bit Field Name	Bit Description
[23]	PSSPADCTRL	PSS State Pad Control This bit controls globally the device IO pad state in power save state as long as SYSC_SPCCFGR:HOLDIO bit is set. This bit functionality is disabled when SYSC_SPCCFGR:HOLDIO bit is '0' and Port Mux directly controls the IO pads. '0': All outputs pads retain their previous value and input buffers will retain their enable or disable state. If the input buffers are enabled, the input level switched to globally defined input level in SYSC_SPCCFGR:GPIL bits '1': All outputs pads are tristated. All input buffers are disabled except for the port pins that are used as external interrupts. SYSC_SPCCFGR:GPIL bits define the input level for the external interrupt port pins
[22]	read0	-
[21:20]	GPIL	Global Pin Input Level Select bit These bits select the programmable input level during power save state as long as SYSC_SPCCFGR:HOLDIO bit is set. When SYSC_SPCCFGR:HOLDIO bit is '0', the Port Mux controls the input level of IOs. GPIL[1:0] = ‘00’ : CMOS hysteresis type A GPIL[1:0] = ‘01’ : Automotive hysteresis GPIL[1:0] = ‘10’ : TTL GPIL[1:0] = ‘11’ : CMOS hysteresis type B **Note:** For SYSC_SPCCFGR:GPIL a pin input level selection should be configured which is supported by all enabled digital inputs. If the configured setting is not supported by a port pin, its digital input will be disabled as soon as SYSC_SPCCFGR:HOLDIO is set to 1. Please refer to the device datasheet for supported pin input levels.
[19:18]	read0	-
[17]	reserved	-
[16]	FCIMEN	Fast Main Clock Input Enable Control bit This bit can be set to support higher frequency clock at Main oscillator input pin. '0': Input clock frequency range for Main oscillator should be with in the range as defined in device specific datasheet. '1': Input clock frequency range for Main oscillator can be much higher. For exact supported frequency range refer to datasheet. Slower clock can also be fed during this mode
[15:13]	read0	-
[12]	PLLINSEL	PLL Input Clock Select Control bit This bit selects input clock for PLL. '0': Main clock is selected as PLL input '1': Reserved for testing This bit should always be written as '0' so that Main oscillator clock is used.
[11]	read0	-

Table 2-102. Special Configuration Register (SYSC_SPCCFGR) bits

Bit Position	Bit Field Name	Bit Description
[10]	GFXSTABS	Graphics PLL Stabilization Time Select Control bit This bit selects the stabilization time for graphics PLL clock. '0': Graphics PLL stabilization time is 2^{14} CLK_MAIN_G '1': Graphics PLL stabilization time is 2^{12} CLK_MAIN_G Graphics PLL stabilization counter starts after Main clock is stable (SYSC_CKSRESTSR:MAINCLKRDY is set) and GFX-PLL is enabled (SYSC_CKSRESTSR:GFXPLLEN = '1'). Once the PLL stabilization time is over, SYSC_CKSRESTSR:GFX-PLLRDY bit will be set.
[9]	SSCGSTABS	SSCG PLL Stabilization Time Select Control bit This bit selects the stabilization time for SSCG PLL clock. '0': SSCG PLL stabilization time is 2^{14} CLK_MAIN_G '1': SSCG PLL stabilization time is 2^{12} CLK_MAIN_G SSCG PLL stabilization counter starts after Main clock is stable (SYSC_CKSRESTSR:MAINCLKRDY is set) and SSCG-PLL is enabled (SYSC_CKSRESTSR:SSCGPLLEN = '1'). Once the PLL stabilization time is over, SYSC_CKSRESTSR:SSCG-PLLRDY bit will be set.
[8]	PLLSTABS	PLL Stabilization Time Select Control bit This bit selects the stabilization time for PLL clock. '0': PLL stabilization time is 2^{14} CLK_MAIN_G '1': PLL stabilization time is 2^{12} CLK_MAIN_G PLL stabilization counter starts after Main clock is stable (SYSC_CKSRESTSR:MAINCLKRDY is set) and PLL is enabled (SYSC_CKSRESTSR:MAINPLLEN = '1'). Once the PLL stabilization time is over, SYSC_CKSRESTSR:MAIN-PLLRDY bit will be set.
[7:2]	read0	-
[1]	FAKEPWRCNT	Fake Power Down Control bit This bit provides the possibility to emulate power domain 2 (PD2) and power domain 3 (PD3) power down scenario while keeping debug/ trace logic functional. User can debug power up operation by using this mode. '0': Normal operation. PD2 and PD3 will be switched OFF as per PSS profile register configuration Power up debugging of PD2 and PD3 is not possible since debug/ trace is also powered down. '1': PD2 and PD3 will not be switched OFF even on applying PSS profile, which says these domains should be switched OFF Trace and debug logic will be fully functional other parts of PD2 and PD3 logic will be under reset.

Table 2-102. Special Configuration Register (SYSC_SPCCFGR) bits

Bit Position	Bit Field Name	Bit Description
[0]	FASTON	Fast On Control bit for Power Controller This bit decides if power domains should be switched ON in parallel or serial fashion. '0': Power controller switch ON each power domain serially (one after another), which takes more time during power switching and it reduces the load on regulator '1': Power controller switch ON all power domains in parallel, which takes less time during power switching and it increases the load on regulator User needs to set this register as per wakeup time requirement and external voltage regulator specification.

2.2.16.2 RC Configuration Register (SYSC_RCCFGR)

RC Configuration Register (SYSC_RCCFGR) has RC clock configuration settings.

RC Configuration Register (SYSC_RCCFGR)

Figure 2-102. RC Configuration Register (SYSC_RCCFGR)

SYSC_RCCFGR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SFREQ	RCTRM[7]	RCTRM[6]	RCTRM[5]	RCTRM[4]	RCTRM[3]	RCTRM[2]	RCTRM[1]	RCTRM[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Table 2-103. RC Configuration Register (SYSC_RCCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	SFREQ	RC Clock Frequency Select bit This bit is used select output frequency generated by RC oscillator. '0': RC clock oscillator configured to generate 8 MHz clock '1': RC clock oscillator configured to generate 12 MHz clock
[7:0]	RCTRM	RC Oscillator Clock Trim This bit is used to trim RC clock frequency. For more details on trim values refer to datasheet.

2.2.16.3 Test 0 Register (SYSC_TESTR0)

Test 0 Register (SYSC_TESTR0) contains test bits reserved for future use.

Test 0 Register (SYSC_TESTR0)

Figure 2-103. Test 0 Register (SYSC_TESTR0)

SYSC_TESTR0																															
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**
TESTBIT[31]	TESTBIT[30]	TESTBIT[29]	TESTBIT[28]	TESTBIT[27]	TESTBIT[26]	TESTBIT[25]	TESTBIT[24]	TESTBIT[23]	TESTBIT[22]	TESTBIT[21]	TESTBIT[20]	TESTBIT[19]	TESTBIT[18]	TESTBIT[17]	TESTBIT[16]	TESTBIT[15]	TESTBIT[14]	TESTBIT[13]	TESTBIT[12]	TESTBIT[11]	TESTBIT[10]	TESTBIT[9]	TESTBIT[8]	TESTBIT[7]	TESTBIT[6]	TESTBIT[5]	TESTBIT[4]	TESTBIT[3]	TESTBIT[2]	TESTBIT[1]	TESTBIT[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-104. Test 0 Register (SYSC_TESTR0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TESTBIT	TESTBIT Test bits. Reserved for future use.

2.2.16.4 Test 1 Register (SYSC_TESTR1)

Test 1 Register (SYSC_TESTR1) contains test bits reserved for future use.

Test 1 Register (SYSC_TESTR1)

Figure 2-104. Test 1 Register (SYSC_TESTR1)

SYSC_TESTR1																															
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**
TESTBIT[31]	TESTBIT[30]	TESTBIT[29]	TESTBIT[28]	TESTBIT[27]	TESTBIT[26]	TESTBIT[25]	TESTBIT[24]	TESTBIT[23]	TESTBIT[22]	TESTBIT[21]	TESTBIT[20]	TESTBIT[19]	TESTBIT[18]	TESTBIT[17]	TESTBIT[16]	TESTBIT[15]	TESTBIT[14]	TESTBIT[13]	TESTBIT[12]	TESTBIT[11]	TESTBIT[10]	TESTBIT[9]	TESTBIT[8]	TESTBIT[7]	TESTBIT[6]	TESTBIT[5]	TESTBIT[4]	TESTBIT[3]	TESTBIT[2]	TESTBIT[1]	TESTBIT[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-105. Test 1 Register (SYSC_TESTR1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TESTBIT	TESTBIT Test bits. Reserved for future use.

2.2.16.5 Test 2 Register (SYSC_TESTR2)

Test 1 Register (SYSC_TESTR2) contains test bits reserved for future use.

Test 2 Register (SYSC_TESTR2)

Figure 2-105. Test 2 Register (SYSC_TESTR2)

SYSC_TESTR2																															
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**
TESTBIT[31]	TESTBIT[30]	TESTBIT[29]	TESTBIT[28]	TESTBIT[27]	TESTBIT[26]	TESTBIT[25]	TESTBIT[24]	TESTBIT[23]	TESTBIT[22]	TESTBIT[21]	TESTBIT[20]	TESTBIT[19]	TESTBIT[18]	TESTBIT[17]	TESTBIT[16]	TESTBIT[15]	TESTBIT[14]	TESTBIT[13]	TESTBIT[12]	TESTBIT[11]	TESTBIT[10]	TESTBIT[9]	TESTBIT[8]	TESTBIT[7]	TESTBIT[6]	TESTBIT[5]	TESTBIT[4]	TESTBIT[3]	TESTBIT[2]	TESTBIT[1]	TESTBIT[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-106. Test 2 Register (SYSC_TESTR2) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TESTBIT	TESTBIT Test bits. Reserved for future use.

2.2.17 Debugger registers

Debugger register (DBGREG) has debugger connection status register, configuration register bits to enable wakeup due to debugger connection, and register to show debugger configuration is done. These registers are initialized on hard reset.

2.2.17.1 JTAG Detect Register (SYSC_JTAGDETECT)

JTAG Detect Register (SYSC_JTAGDETECT) has Debugger connection status. Debugger connection to the device is detected by counting the fixed number of debugger clock after nTRST rising edge.

JTAG Detect Register (SYSC_JTAGDETECT)

Figure 2-106. JTAG Detect Register (SYSC_JTAGDETECT)

SYSC_JTAGDETECT																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGCON
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-107. JTAG Detect Register (SYSC_JTAGDETECT) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	DBGCON	Debugger Connected Status showing debugger is connected to the device. '0': Debugger unconnected '1': Debugger connected When this bit is '1' it overrides the clock enable bit of the Debug and Trace module (SYSC_APPCKER:ENTRACEPD2 and SYSC_APPCKER:ENTRACEPD3), when the device is in RUN state or when the fake power mode is selected. Causes debugger and trace clock enabling, even if they are switched OFF in profile.

2.2.17.2 JTAG Configuration Register (SYSC_JTAGCNFG)

JTAG Configuration Register (SYSC_JTAGCNFG) has configuration bit to show that coresignt configuration is completed. This register is initialized to ‘0’, when debugger is connected. This bit is set by the debugger after the debugger configuration is done. This register bit is read by BootROM to check the status of debugger configuration.

JTAG Configuration Register (SYSC_JTAGCNFG)

Figure 2-107. JTAG Configuration Register (SYSC_JTAGCNFG)

SYSC_JTAGCNFG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGDONE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 2-108. JTAG Configuration Register (SYSC_JTAGCNFG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	DBGDONE	Debugger Configuration Done This register bit is written by AHB master to indicate that debugger configuration is done. '0': Debugger configuration ongoing '1': Debugger not connected or debugger configuration is done

2.2.17.3 JTAG Wakeup Register (SYSC_JTAGWAKEUP)

JTAG Wakeup Register (SYSC_JTAGWAKEUP) has configuration bit to enable wakeup on debugger connection.

JTAG Wakeup Register (SYSC_JTAGWAKEUP)

Figure 2-108. JTAG Wakeup Register (SYSC_JTAGWAKEUP)

SYSC_JTAGWAKEUP																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGWKEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 2-109. JTAG Wakeup Register (SYSC_JTAGWAKEUP) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	DBGWKEN	Debugger Wakeup Enable This bit is used to enable wakeup by debugger whenever debugger is connected to the device. This bit can be written by the debugger to mask wakeup due to debugger connection. '0': Wakeup due to debugger connection is masked '1': Wakeup due to debugger connection is enabled **Note:** If debugger is already connected and this bit is set then a wakeup is triggered as soon as device goes in PSS. Means if user wants to go in PSS state then this bit should be written as '0'. But in case if this bit is '0' and device is in the PSS state then a fresh debugger connection will not wakeup the device and user should use some external event to wakeup the device.

2.3 Operation of System Controller

This chapter describes the operation of the System Controller in detail.

2.3.1 Device mode

Below modes of operation are supported by the device.

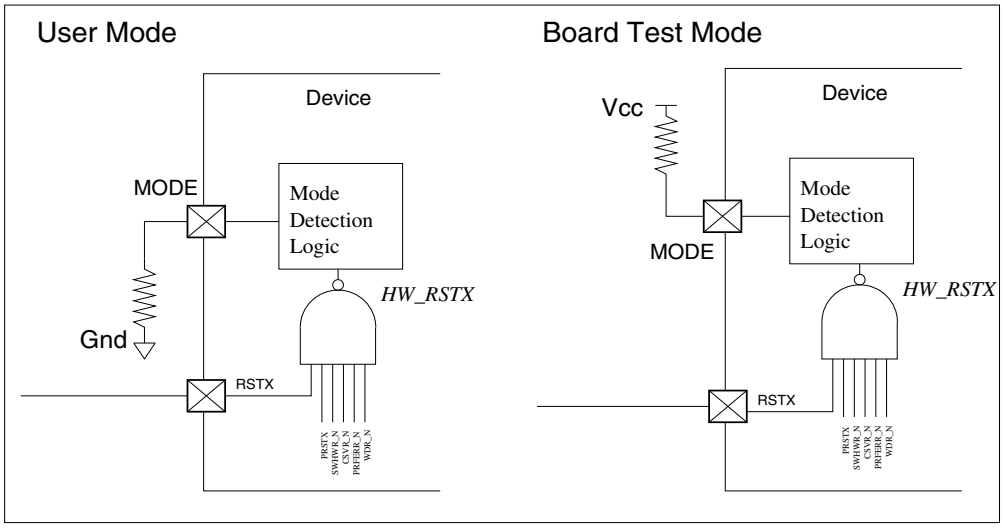
- User mode
- Board test mode
- Boundary scan mode

The MODE pin is used to select the user and board test mode. The BootROM program reads the status of the MODE pin and changes the device mode accordingly.The bounday scan mode is entered by the DAP/TAP.

2.3.1.1 MODE selection

The MODE pin defines the operation mode of the MCU at the time of reset release. The MODE pin setting is internally sampled at power on reset (PRSTX), external reset (RSTX), clock supervisor reset (CSVR_N), software-triggered hardware reset (SWHWR_N), watchdog reset (WDR_N), and profile error reset (PREFRR_N). For more information of the reset sources refer to 2.3.3.1 Reset sources Figure 2-109 shows setup for device mode change at the time of reset.

Figure 2-109. Setup for changing device mode



2.3.1.2 User mode

To enter user mode the MODE pin must be asserted LOW during reset release. For device MODE pin setting to enter into user mode refer to Figure 2-109.

In user mode the device executes user applications on the integrated core. In this, mode device states can be changed, in order to save power. For more details refer to 2.3.2 Device state.

In this mode the user application starts from internal Flash/ROM. For details on device booting refer to 15. Boot ROM Hardware Interface. Device security is active in this mode based on SDR configuration. For more details on device security refer to 5. Security Checker.

2.3.1.3 Board test mode

To enter into board test mode, the MODE pin must be asserted HIGH during reset release. For device MODE pin setting to enter into board test mode refer to Figure 2-109. The following board test mode is supported:

- Flash parallel programming mode

Device security is active in this mode based on SDR configuration. For more details on device security refer to 5. Security Checker.

2.3.1.4 Boundary scan mode

Boundary scan mode is special mode in the device. The boundary scan mode is entered using DAP. Only then the TAP gets connected to the JTAG pins and can be used for boundary scan. Device security is active in this mode based on SDR configuration. For more details on device security refer to 5. Security Checker. To exit this mode nTRST and RSTX should be asserted simultaneously.

2.3.2 Device state

The MCU operates in one of the following two states at any point of time:

- RUN state
- Power Saving State

RUN state is the state where CPU (Arm Cortex-R4) is powered ON and CPU clock is active; whereas Power Saving State (henceforth called as PSS) is the state where CPU is not operational. The user can choose the clock and power configuration for each state. This will define the performance and current consumption of the device. This set of configuration registers are called profile registers.

There are two profiles defined for the device:

- RUN profile
- PSS profile

These profiles configure the device settings, based on the state the device is in. For more details on profiles refer to 2.3.2.7 Device state profiles.

2.3.2.1 RUN state

The RUN state is defined by the RUN profile configured by the software. This is the default state of the device after being powered ON and after all hard reset assertions. For hard reset sources refer to 2.3.3.1.The device also enters RUN state from PSS on wakeup event. The RUN state has CPU powered ON and CPU clock is active, which means Interrupt Controller, HPM, Retention RAM, TCFLASH and EEPROM FLASH, BootROM, and System Controller are provided with clock and power. The state (clock and power settings) of the rest of the system can be chosen by the user by RUN profile configuration. For more details on the configurable RUN profile refer to 2.3.2.10 Device state table. Switching from one operating point to other within RUN state i.e. RUN to RUN switching or from RUN to PSS switching can only be initiated by the software.

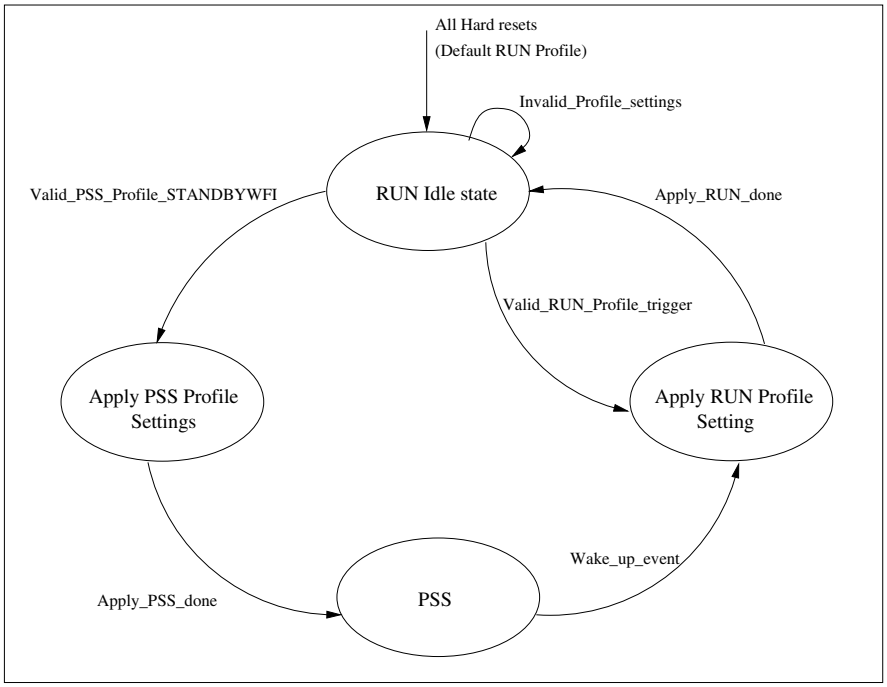
2.3.2.2 Power Saving State (PSS)

PSS is entered from RUN state under software control. In this state CPU is non-operational, which means either clock to CPU is switched OFF or CPU clock and CPU power domain are switched OFF. Interrupt Controller, BootROM, TPU and EEFLASH have also their clocks cut and hence non- operational. The state of the remaining part of the MCU depends on the PSS profile, configured by the software. For the configurable PSS profile settings refer to 2.3.2.10 Device state table. In this state, user can configure to stop all the clocks and switch OFF all switchable power domains, in order to minimize the power consumption of the device to the lowest possible. The device can exit from this state to the RUN state by wakeup event. For all possible wakeup sources refer to 2.3.2.6 PSS to RUN state switching.

2.3.2.3 Device state switching

Figure 2-110 gives the top level overview of device state switching.

Figure 2-110. Device state switching



Note: 'Apply RUN profile settings' and 'Apply PSS profile settings' have more states in turn. For more details refer to 2.3.2.4 RUN to RUN state switching, 2.3.2.5 RUN to PSS switching, and 2.3.2.6 PSS to RUN state switching.

2.3.2.4 RUN to RUN state switching

Device state logic supports device switching from one operating point to the other, based on the current and performance need. This state switching is done as explained in the following sequence:

1. When user wants to switch to new operating point depending upon the use case, user needs to configure the new RUN profile.
2. New RUN profile configuration can be applied to System Controller by writing 0xAB into Trigger Register SYSC_TRGRUNCNTR:APPLYRUN of RUN Profile. This triggers below sequences in device state control logic.
3. New RUN profile is checked for validity. If the validity check fails (for valid settings of RUN profile refer to 2.3.2.8 RUN profile) for the profile, error flag SYSC_SYSERRR:RUNERRIF is set and error interrupt is generated. The new RUN profile is ignored and the device stays in the current RUN State with current profile settings unaltered.
4. If the profile settings are valid, the device applies the new settings in stages, as below.
   - Device busy flag SYSC_SYSSTSR:RUNBUSY is set
   - RUN profile settings are copied into APPLIED profile register set and APPLIED profile register set is locked by the hardware till the new RUN profile is applied completely to the device
   - All new source clocks, which are configured to be enabled in the profile are started. The corresponding CSV enable settings are applied. LVD settings are applied. Device state logic goes to the next stage, only when all the source clocks enabled are stabilized. The availability of the source clocks is also updated in Clock Source Enable Status Register (SYSC_CKSRESTSR) in profile status register group
   - All new power domain configuration, clock selection, clock divider, and clock enable are applied. The System Controller waits here till the clock switching is done i.e. all power OFF and power ON request of switchable power domain is complete. The power ON request of switchable power domain is complete with power stabilization and initialization to reset value

- All source clocks which are configured to be disabled are stopped. The corresponding CSV disable settings are applied
- Device enters into RUN state with new RUN profile settings
- Device busy flag SYSC_SYSSTSR:RUNBUSY is cleared by the device state control logic. RUN profile done flag in System Status Register (SYSC_SYSSTSR:RUNDN) is set. If the RUN profile applied done interrupt enable bit (SYSC_SYSINTER:RUNDNIE) is set, device status interrupt is generated

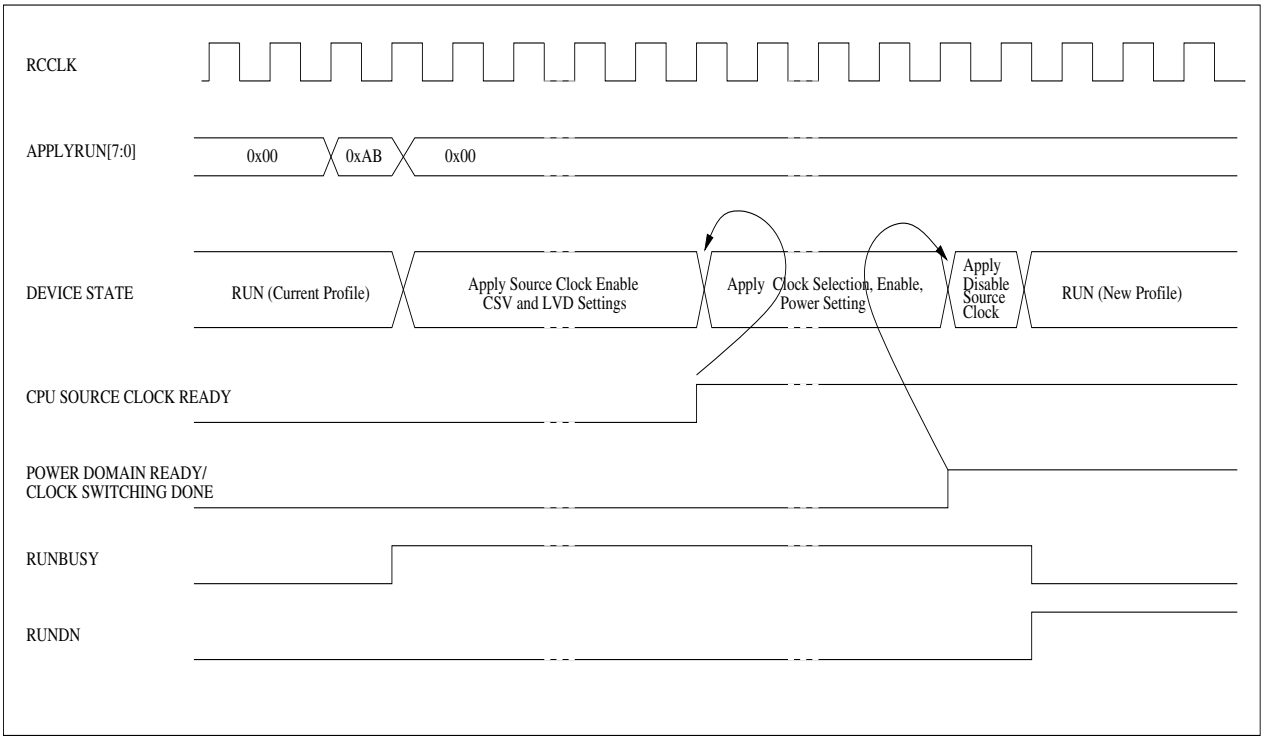
This application of the RUN profile in above stages ensures device safety and the device will never switch to a clock which is not available or not stable.

The time taken by the device state logic to apply the new RUN profile setting will depend upon the profile settings. Switching the CPU clock to new requested clock, after the new clock is stabilized, ensures that the CPU will be operational during this state switching with current clock setting.

When device state control logic is applying the settings, user can configure new RUN profile settings. However, applying the RUN trigger when the state control logic is busy in applying the current profile setting (when SYSC_SYSSTSR:RUNBUSY is '1'), sets the error flag SYSC_SYSERRR:TRGERRIF and this new profile settings are ignored. However, this does not affect the current profile setting of the application and the settings which are currently being applied will continue to get applied.

Note: The clock switching of the respective clock domain happens, only after the corresponding source clock stabilization.

Figure 2-111. RUN to RUN switch sequence



2.3.2.5 RUN to PSS switching

Device can enter into PSS from RUN state under user control. This is done in below sequence.

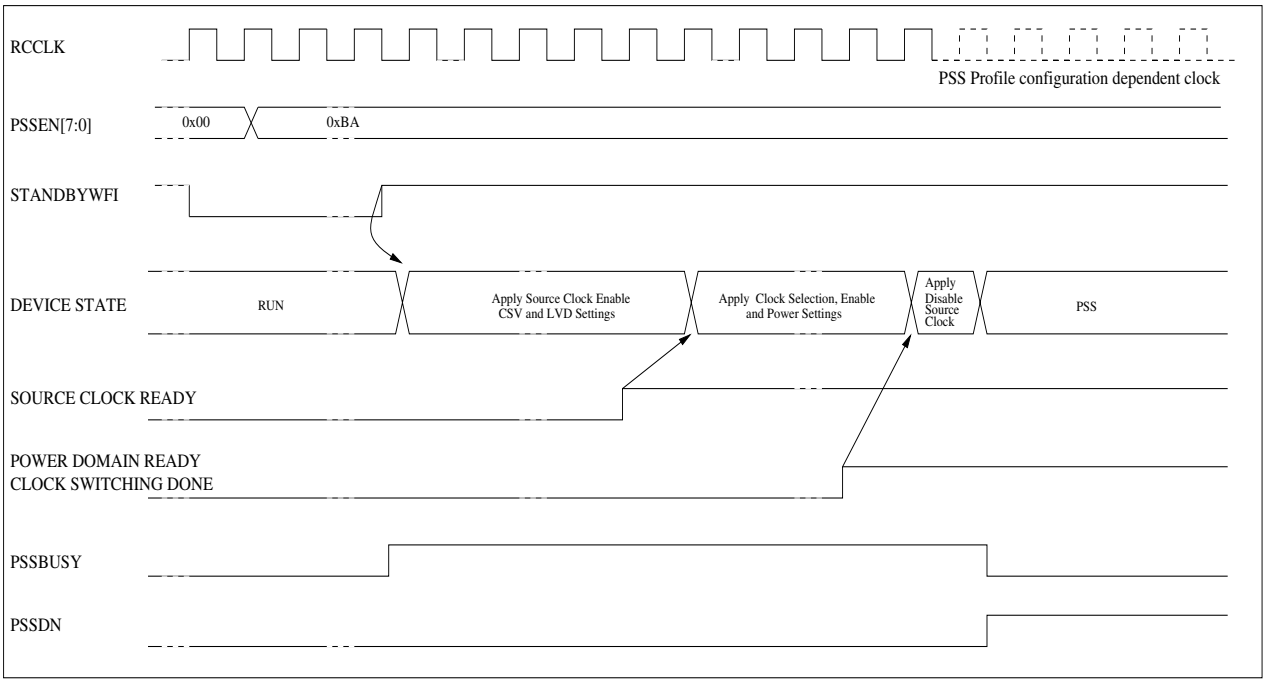
1. User configures the PSS profile for new PSS by applying protected sequence. The user also needs to configure the RUN profile to be applied after wakeup.
2. PSS switching is enabled by writing 0xBA into PSS Enable Register (SYSC_PSSENR:PSSEN). Both RUN and PSS profiles are checked for consistency. If the validity check fails for the RUN profile, then the error flag SYSC_SYSERRR:RUNWKERRIF bit is set. The RUN profile check ensures, that device wakes up with the right RUN profile settings. If the PSS profile fails for validity check, the error flag SYSC_SYSERRR:PSSERRIF is set. These error flags causes NMI, since these interrupt cannot be disabled. PSS Enable Register is cleared by the hardware as soon as any write access happens in RUN or PSS profile. This avoids profile corruption after validity check. If validity check fails then the RUN to PSS sequence needs to be restarted, if the user wants to enter into PSS. If the validity check passes, state switching continues once WFI instruction is executed.
3. On WFI instruction execution in CPU, CPU enters into standby mode and triggers PSS switching in state control logic. If an Interrupt is pending during a WFI instruction, neither the PSS profile nor a new RUN profile (after wake-up) is applied. However, the PSS entry stays enabled until the next WFI instruction (without a pending interrupt).
4. Both RUN and PSS profiles are checked for consistency. If the validity check fails for RUN profile, then error flag SYSC_SYSERRR:RUNWKERRIF bit is set. The RUN profile check ensures, that device wakes up with the right RUN profile settings. If PSS profile fails for validity check, error flag SYSC_SYSERRR:PSSERRIF is set. The new PSS profile is ignored and the device stays in the current RUN state with current profile settings.The validity check is performed in one RC clock cycle.
5. If the profile settings are valid, the device applies the new settings in stages, as below:
   - Device busy flag SYSC_SYSSTSR:PSSBUSY is set
   - PSS profile setting is copied into APPLIED profile register set and APPLIED profile register set is locked
   - All new source clock which are configured to be enabled in the profile are started. The corresponding CSV enable settings are applied. LVD settings are applied. Device state logic goes to the next stage, only when all the source clocks enabled are stabilized. The availability of the source clock is also updated in Clock Source Enable Status Register (SYSC_CKSRESTSR) in profile status register group
   - All new power domain configuration, clock selection, clock divider, and clock enable are applied. System Controller waits here till the clocks switching is done i.e. all power OFF and power ON request of switchable power domain is complete. All power ON request of switchable power domain is complete with power stabilization and initialization to reset value
   - All source clocks which are configured to be disabled are stopped. The corresponding CSV disable settings are applied
   - Device busy flag SYSC_SYSSTSR:PSSBUSY is reset. PSS profile done flag in System Status Register (SYSC_SYSSTSR:PSSDN) is set
   - Device enters into PSS

This application of the PSS profile in above stages ensures device safety i.e. the device will never switch to the clock which is not available or not stable.

The time taken by the device state logic to apply the PSS profile setting depends upon the profile settings.

Note: The RUN to PSS switching will be slower, if new clock sources are enabled during PSS switching.

Figure 2-112. RUN to PSS switching sequence



2.3.2.6 PSS to RUN state switching

PSS to RUN state switching is triggered by the hardware wakeup event. The following wakeup source triggers the device to enter into RUN state from PSS:

- All device interrupts
- NMI
- Hard resets
- JTAG wakeup

Refer to the device datasheet for all the interrupt sources in the device and NMI sources. For the hard reset sources refer to 2.3.3.1 Reset sources.

Connection of the debugger to the device causes JTAG wakeup event.The wakeup event due to hard reset brings the device into RUN state with default RUN profile. For default RUN profile refer to Table 2-110.

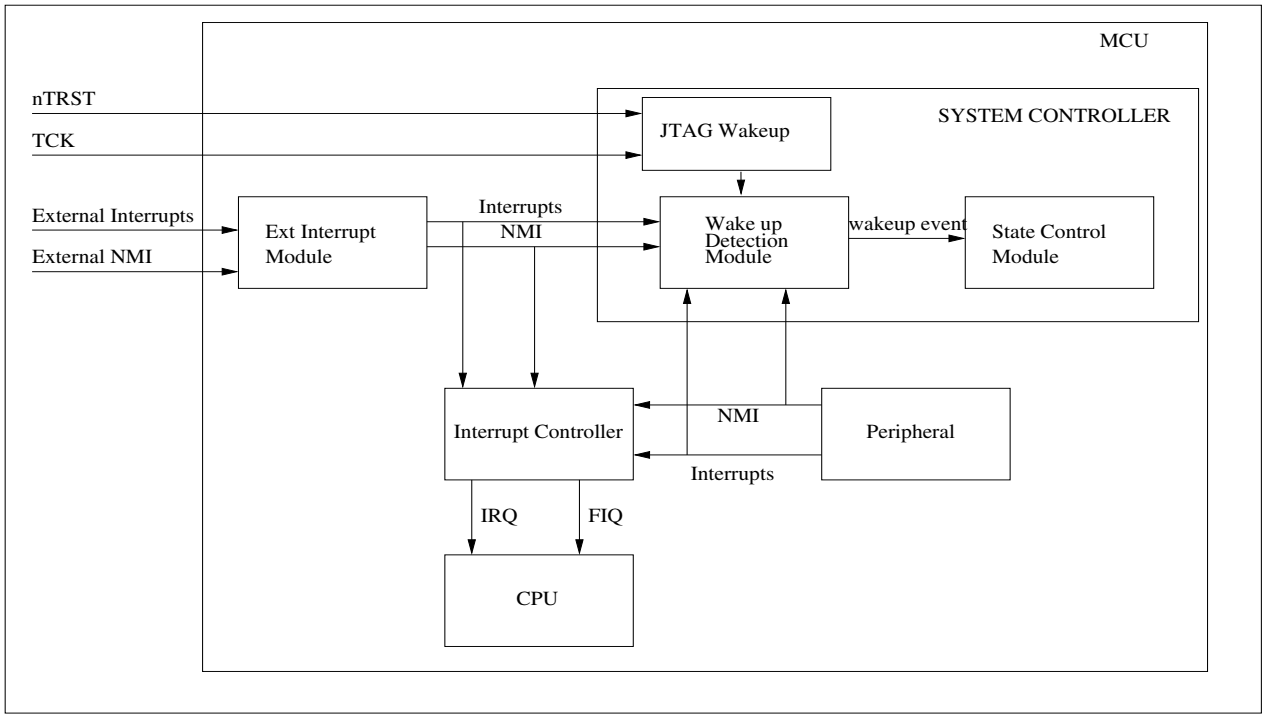
Other wakeup events i.e. device interrupts, NMI, and JTAG wakeup will bring the device into RUN state with user pre-configured RUN profile.

Note: Debugger can cause software reset, when the device is in PSS. But this does not cause change in the state of the device. At the same time this reset, resets the CPU and other peripheral logic part in MCU. For more details on the parts of the MCU which gets reset with soft reset refer to Table 2-111.

Wakeup Event detection

Figure 2-113 shows sources of wakeup and wakeup controller interconnection.

Figure 2-113. Wakeup detection

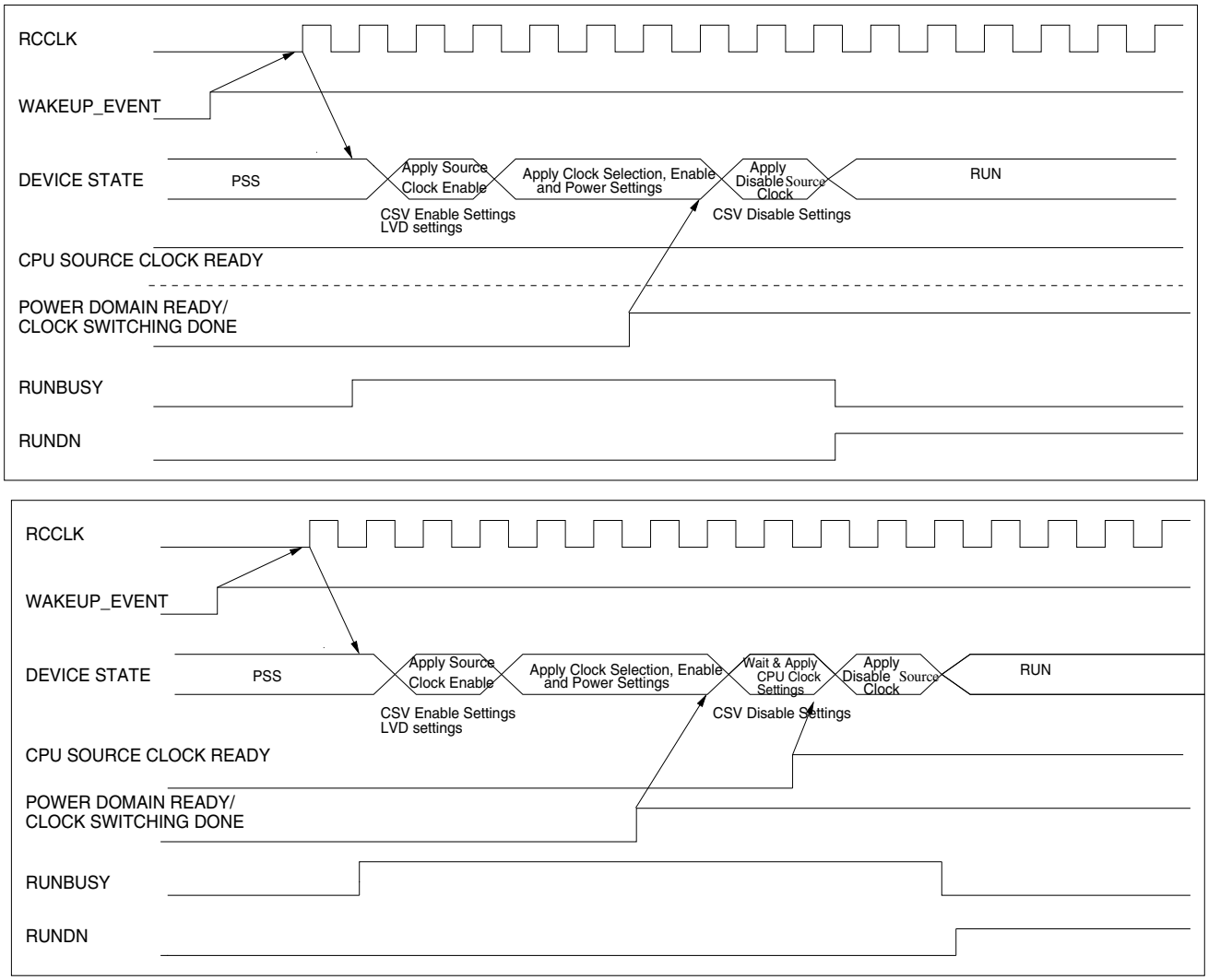


The wakeup event due to NMI, Internal and External Interrupts and JTAG wakeup triggers below sequence:

1. RUN profile is checked for consistency. Profile error reset (PRFERR_N) is generated if the consistency check fails because of bit-flipping, or modification of RUN profile by debugger in PSS. The device enters into RUN state with the default RUN profile settings.
2. If the RC clock is switched OFF in PSS, wakeup event switches ON RC clock, which is required for state control module operation. System Controller waits for RC clock stabilization in this state. This step is skipped if RC clock is already ON.
3. All new source clocks which are configured to be enabled in the profile are started. The corresponding CSV enable settings are applied. LVD settings are applied. The availability of the source clock is also updated in Clock Source Enable Status Register (SYSC_CKSRESTSR) in profile status register group.
4. All new power domain configuration, clock selection, clock divider, and clock enable are applied. If the CPU clock selected in the profile is available at this stage, this clock is selected for CPU, otherwise RC clock is selected as the CPU clock. This ensures the reliable wakeup of the CPU. System Controller waits here till the clock switching is done, all power OFF and power ON request of switchable power domains is complete. The power ON request of switchable power domains is complete with power stabilization and initialization to reset value
5. The device state logic waits for the CPU clock selected by the profile is stabilized. When this clock is stabilized, the CPU clock setting with this new stabilized clock is applied. This state is skipped, if the CPU clock selected by the RUN profile is already applied in the previous state.
6. All source clocks which are configured as disabled are stopped. The corresponding CSV disable settings are applied.
7. Device enters in RUN state.

Figure 2-114 shows this state switching sequence. The first figure shows the sequence wherein the CPU clock at the wakeup is not switched OFF in PSS. The second figure shows the sequence where CPU clock at the wakeup is switched OFF in PSS.

Figure 2-114. PSS to RUN state switching



Note: For the wakeup sequence due to hard reset, refer to Figure 2-115.

2.3.2.7 *Device state profiles*

The profile defines the device operating point. The profile consists of the below configuration settings:

- All source clock settings of the device i.e. RC oscillator, Main oscillator, Sub oscillator settings and all PLL settings
- Clock selection, clock divider, and clock distribution settings
- Power domain configuration settings
- CSV settings
- LVD Settings

Two profiles i.e. RUN profile and PSS profile are defined corresponding to the device RUN state and PSS state.

For profile register set refer to 2.2 System Controller registers.

2.3.2.8 RUN profile

The RUN state of the device is characterized by this profile. This profile has the configuration settings outlined below fixed, which means software cannot overrule these settings. These settings are always fixed for RUN profile.

- Core power domain (PD3) and peripheral power domain (PD2) are ON
- Core group clock is ON
- HPM clock is ON
- Internal memory group clock is ON

The rest of the settings can be freely chosen by the user, but they need to be valid. Refer to Table 2-110 for the configurations, which are fixed and which are configurable in the RUN profile.

Invalid RUN profile settings for RUN to RUN transitions

The following RUN profile settings are illegal and will set the SYSC_SYSSTSR:IRPARUN flag:

- Request that the clock is enabled but disable the corresponding Power Domain i.e. PDx = ‘0’ and ENCLKPDx = ‘1’ in the RUN profile
- Nested power domain rule violation, i.e. request that PD2 is switched OFF and that PD3 and PD5 are switched ON in the RUN profile. Refer to 2.3.4.2 Power domain control for more details
- Selecting a source clock for distribution which is not enabled in the RUN profile
- Disabling the main clock when any of the PLLs are ON in the RUN profile
- The Clock Supervisor is enabled and at the same time, the reference clock or monitor clock is disabled in the RUN profile. Or disabling the Clock Supervisor (CSV is enabled in the APP profile and disabled in thr RUN profile) when monitored clocks are enabled in both the APP and RUN profiles
- Changing of the PLL settings (a PLL setting differs between APP and RUN profiles) while the PLL is ON (i.e. while the PLL is enabled in both APP and RUN profiles). This check does not apply to the DIVM setting of the PLL
- Disabling the source clock in the RUN profile which is currently used (i.e. the Watchdog is enabled in the device RUN state and corresponding clock is selected) by the watchdog module for watchdog operation
- iRUN_SYSCSL must not be set to “No Clock select” in the RUN profile

Invalid RUN profile settings for RUN to PSS transitions

The following RUN profile settings for wakeup are illegal and will set the SYSC_SYSSTSR:IRPAPSS flag:

- Request that the clock is enabled but disable the corresponding Power Domain i.e. PDx = ‘0’ and ENCLKPDx = ‘1’ in the RUN profile
- Nested power domain rule violation, i.e. request that PD2 is switched OFF and that PD3 and PD5 are switched ON in the RUN profile. Refer to 2.3.4.2 Power domain control for more details
- Selecting a source clock for distribution which is not enabled in the RUN profile
- Disabling the main clock when any of the PLLs are ON in the RUN profile
- The Clock Supervisor is enabled and at the same time, the reference clock or monitor clock is disabled in the RUN profile. Or disabling the Clock Supervisor (CSV is enabled in the APP profile and disabled in thr RUN profile) when monitored clocks are enabled in both the APP and RUN profiles
- Changing of the PLL settings (a PLL setting differs between PSS and RUN profiles) while the PLL is ON (i.e. while the PLL is enabled in both PSS and RUN profiles). This check does not apply to the DIVM setting of the PLL
- The PSS profile (see 2.3.2.9 PSS profile) has a PD2 “switch OFF” request and the RUN profile has the “RC clock not chosen as system clock”. This check is required for device security as the security key needs to be re-initialized in this case with the “RC clock as syatem clock”
- Disabling the source clock in the RUN profile which is currently used (i.e. the Watchdog is enabled in the device RUN state and corresponding clock is selected) by the watchdog module for watchdog operation
- iRUN_SYSCSL must not be set to “No Clock select” in the RUN profile

2.3.2.9 PSS profile

The Power Saving State (PSS) of the device is characterized by this profile. The configuration of this profile (described below) is fixed, which means software cannot overrule these settings. The setting below is hardwired for the PSS profile.

- Core group clock is OFF

The rest of the settings can be freely chosen by the user but they need to be valid. For the configuration settings which are fixed and which are configurable in the PSS profile, refer to Table 2-110.

Invalid PSS profile settings for RUN to PSS transitions

The following PSS profile settings are illegal and will set the SYSC_SYSSTSR:IPPAPSS flag:

- Request that the clock is enabled but disable the corresponding Power Domain i.e. PDx = ‘0’ and ENCLKPDx = ‘1’ in the PSS profile
- Nested power domain rule violation, i.e. request that PD2 is switched OFF and that PD3 and PD5 are switched ON in the PSS profile. Refer to 2.3.4.2 Power domain control for more details
- Selecting a source clock for distribution which is not enabled in the PSS profile
- Disabling the main clock when any of the PLLs are ON in the PSS profile
- The Clock Supervisor is enabled and at the same time, the reference clock or monitor clock is disabled in the PSS profile. Or disabling the Clock Supervisor (CSV is enabled in the APP profile and disabled in the PSS profile) when monitored clocks are enabled in both the APP and PSS profiles
- Changing of the PLL settings (a PLL setting differs between APP and PSS profiles) while the PLL is ON (i.e. while the PLL is enabled in both APP and PSS profiles). This check does not apply to the DIVM setting of the PLL
- Disabling the source clock in the PSS profile which is currently used (i.e. the Watchdog is enabled in the device PSS state and corresponding clock is selected) by the watchdog module for watchdog operation

Note: if the WDG_CFG:ALLOWSTOPCLK bit is set in watchdog module and the device is entering deep PSS where all the source clocks are cut, then this check is omitted. For more details on WDG_CFG:ALLOWSTOPCLK configuration, refer to 4. Watchdog Timer.

- An LVD is switched on (i.e. the LVD is off in APP profile and on in PSS profile) or an LVD threshold is changed (i.e. the LVD threshold differs between APP and PSS profiles) when the RC oscillator is disabled in the PSS profile. This check is necessary to make sure the RC clock is available for the stabilization time of the LVD
- Enabling a low voltage detector (i.e PSSLVDCFGR:LVDEx = ‘1’ and APPLVDCFGR: LVDEx=’0’) if the RC oscillator is disabled in PSS state
- Changing the configuration of a low voltage detector if the RC oscillator is disabled in PSS state.

2.3.2.10 Device state table

Table 2-110 shows the device state in RUN state and PSS. For more details on configuration parameters a- State of the device on hard reset refer to 2.3.1 Device mode and 2.2.5 PSS profile register set.

Table 2-110. Device state table

Configuration setting	Default RUN configuration[a]	RUN configuration	PSS configuration
Always ON power domain (PD1)			
CLK_CFG_PD1	ON	ON	Programmable
Peripheral power domain (PD2)			
PD2 power switch	ON	ON	Programmable[b]
CLK_DBG_PD2	ON	Programmable[c]	Programmable[c]
CLK_TRACE_PD2	ON	Programmable[c]	Programmable[c]
CLK_HPM_PD2	ON	ON	Programmable[c]
CLK_DMA_PD2	ON	Programmable[c]	Programmable[c]
CLK_PERI0_PD2	ON	Programmable[c]	Programmable[c]
CLK_PERI1_PD2	ON	Programmable[c]	Programmable[c]
CLK_PERI3_PD2	ON	Programmable[c]	Programmable[c]
CLK_PERI4_PD2	ON	Programmable[c]	Programmable[c]
Core power domain (PD3)			
PD3 power switch	ON	ON	Programmable
CLK_SYS_PD3	ON	ON	OFF
CLK_DBG_PD3	ON	Programmable[c]	Programmable[c]
CLK_TRACE_PD3	ON	Programmable[c]	Programmable[c]
CLK_HPM_PD3	ON	Programmable[c]	Programmable[c]
CLK_MEM_I_PD3	ON	ON	OFF
CLK_MEM_E_PD3	ON	Programmable[c]	Programmable[c]
CLK_SPI_PD3	ON	Programmable[c]	Programmable[c]
Retention power domain (PD4)			
PD4 power switch	ON	Programmable	Programmable
CLK_CFG_PD4	ON	Programmable[c]	Programmable[c]
Graphics power domain (PD5)			
PD5 power switch	ON	Programmable	Programmable
CLK_GFX_PD5	ON	Programmable[c]	Programmable[c]
CLK_PIX_PD5	ON	Programmable[c]	Programmable[c]
CLK_SPI_PD5	ON	Programmable[c]	Programmable[c]
Source clock			
SRC oscillator	ON	Programmable[d]	Programmable[d]

Table 2-110. Device state table

Configuration setting	Default RUN configuration[a]	RUN configuration	PSS configuration
RC oscillator	ON	ON	Programmable[d]
Main oscillator	ON	Programmable[d]	Programmable[d]
Sub oscillator	ON	Programmable[d]	Programmable[d]
Main PLL	OFF	Programmable[d]	Programmable[d]
SSCG PLL	OFF	Programmable[d]	Programmable[d]
GFX PLL	OFF	Programmable[d]	Programmable[d]
Clock mux settings			
System clock select	RC clock	Programmable[e]	Programmable
Peripheral group 0 clock select	RC clock	Programmable	Programmable
Peripheral group 1 clock select	RC clock	Programmable	Programmable
Peripheral group 3 clock select	RC clock	Programmable	Programmable
Pixel clock select for Graphics subsystem[f]	PLL clock	Programmable	Programmable
SPI clock select for Graphics Subsystem	Main PLL clock	Programmable	Programmable
Watchdog clock select	RC clock	Programmable[g]	Programmable[g]
Clock division settings			
System clock division value	1	Programmable	Programmable
Trace clock division value	1	Programmable	Programmable
HPM clock division value	1	Programmable	Programmable
External bus clock division value	1	Programmable	Programmable
External memory clock division value	1	Programmable	Programmable
Configuration clock division value	1	Programmable	Programmable
Peripheral group 4 clock division value	1	Programmable	Programmable
Graphics clock division value	1	Programmable	Programmable
Peripheral group 0 clock division value	1	Programmable	Programmable
Peripheral group 1 clock division value	1	Programmable	Programmable
Peripheral group 3 clock division value	1	Programmable	Programmable
Clock output function division value	1	Programmable	Programmable
LVD settings			
Low voltage detector enable for 5.0 V	Enabled	Programmable	Programmable
Low voltage detector enable for 3.3 V	Disabled	Programmable	Programmable
Low voltage detector enable for 1.2 V	Enabled	Programmable	Programmable
Reference voltage selection for 5.0 V LVD	2.7 V	Programmable	Programmable
Reference voltage selection for 3.3 V LVD	2.2 V	Programmable	Programmable
Reference voltage selection for 1.2 V LVD	0.9 V	Programmable	Programmable
Main PLL settings			
PLLDIVL	0x0	Programmable[h]	Programmable[h]
PLLDIVM	0x1	Programmable	Programmable

Table 2-110. Device state table

Configuration setting	Default RUN configuration[a]	RUN configuration	PSS configuration
PLLDIVN	0x0D	Programmable[h]	Programmable[h]
SSCG PLL settings			
SSCGDIVL	0x0	Programmable[h]	Programmable[h]
SSCGDIVM	0x1	Programmable	Programmable
SSCGDIVN	0x0D	Programmable[h]	Programmable[h]
SSCGDIVP	0x01	Programmable[h]	Programmable[h]
SSCGRATE	0x29	Programmable[h]	Programmable[h]
SSCGMODE	0	Programmable[h]	Programmable[h]
SSCGFREQ	0	Programmable[h]	Programmable[h]
SSCGSSEN	0	Programmable[h]	Programmable[h]
GFX PLL settings			
GFXDIVL	0	Programmable[h]	Programmable[h]
GFXDIVN	0x0D	Programmable[h]	Programmable[h]
GFXDIVP	0x01	Programmable[h]	Programmable[h]
GFXRATE	0x29	Programmable[h]	Programmable[h]
GFXMODE	0	Programmable[h]	Programmable[h]
GFXFREQ	0	Programmable[h]	Programmable[h]
GFXSSEN	0	Programmable[h]	Programmable[h]
CSV settings			
Main oscillator clock supervisor enable	Disabled	Programmable[i]	Programmable[i]
Sub oscillator clock supervisor enable	Disabled	Programmable[i]	Programmable[i]
Main PLL clock supervisor enable	Disabled	Programmable[i]	Programmable[i]
SSCG PLL clock supervisor enable	Disabled	Programmable[i]	Programmable[i]
GFX PLL clock supervisor enable	Disabled	Programmable[i]	Programmable[i]

a. State of the device on hard reset.
b. PD2 power domain must be ON if PD3 or PD5 is ON.
c. Programmable clock can be enabled only if corresponding power domain is ON.
d. Source clock can only be switched OFF, when none of the clock select mux use this source clock.
e. During PSS entry, if PSS profile has PD2 switch OFF request, then system clock select needs to be set to RC clock.
f. In case PIXCSL selects external clock (CLK_EXT_GFX), it should be available to device. This is not taken care in device profie check.
g. Watchdog clock select programming is done by configuring WDG_CFG:CLKSEL bit in the watchdog module. For more details refer to 4. Watchdog Timer
h. These PLL settings can be programmed only when PLL is disabled.
i. CSV can be enabled with enabling the monitoring clock or when monitored clocks are already enabled. CSV can be disabled, only when the monitored clocks are disabled.

2.3.3 Reset and power up

This device has four types of reset.

1. Hard reset group, which resets complete device and starts new mode and security evaluation.
2. Soft reset, which resets core and peripheral domain. All mode, security and clock settings are not changed by soft reset.
3. Power domain reset, generated by power controller, which resets only respective power domains.
4. JTAG reset (nTRST) and Debug & Trace reset (DBGR_N), which resets TAP and Debug & Trace module respectively.

All reset sources (except nTRST and DBGR_N) are stored in the Reset Cause Register, which can be read to find the reason for reset cause. This register needs to be cleared before next reset, otherwise it will also show previous reset cause along with the latest one.

Power domain 2 reset triggers reevaluation of security settings.

2.3.3.1 Reset sources

This section provides description about all reset sources.

2.3.3.2 Hard reset

Hard reset, resets complete device except TAP and starts new mode and security evaluation. This group of reset consists of power reset, external reset (RSTX), software-triggered hard reset (SWHWR), Clock Supervisor resets, profile error reset, and watchdog reset. Description of each hard reset can be found below.

PRSTX (power reset)

The power reset (PRSTX) is a combination of the low voltage reset on 5 V, 3.3 V, and 1.2 V supply. If any of the voltage supplies are going lower than the threshold value, device will go to reset. Power reset resets the whole device once voltage is stable and the low voltage reset extension time has expired, power reset is deasserted, and device starts booting from Boot-ROM. SYSC_RSTCAUSEBT:PRSTX and SYSC_RSTCAUSEUR:PRSTX register bits are set whenever PRSTX is encountered.

RSTX (external reset)

External reset is generated by low level input to the external pin RSTX. There is a Noise Filter to avoid device reset due to high frequency noise. RSTX resets complete device including mode detection, security setting and debug/trace system. The debug/trace system is not reset through this reset if debugger is connected.

Note: External reset must be asserted until external voltage supply is stable.

The external voltage supply 5 V, 3.3 V, and 1.2 V should be applied in sequence, where 5 V and 3.3 V applied first and then 1.2 V is applied.

SWHWR_N (software-triggered hard reset)

Software-triggered hard reset can be achieved by writing 0xA5 in SYSC_RSTCNTR:SWHRST register. This reset provides user to trigger a new mode evaluation and a new security evaluation through the software. SWHWR_N resets complete device including mode detection, security setting, and debug/trace system. The debug/trace system is not reset through this reset if debugger is connected.

CSVR_N (Clock Supervisor reset)

Clock Supervisor reset is generated when the Main oscillator clock, Sub oscillator clock or any of the PLL clocks is missing/unstable and it is being used in the device. Once this reset is asserted default RUN profile is applied and all clocks in the device switches to CLK_RC. The debug/trace system is not reset through this reset if debugger is connected. For more details refer to 2.3.7 Clock Supervisor.

PRFERR_N (profile error reset)

Profile error reset (PRFERR_N) is generated by the device on invalid RUN profile, when device wakes up from PSS. For more details refer to 2.3.2.6 PSS to RUN state switching. This reset is to take care of any hardware error (like bit-flipping) during PSS. The debug/trace system is not reset through this reset if debugger is connected.

WDR_N (watchdog reset)

The watchdog reset occurs, if the Watchdog is not triggered correctly in the programmed time interval. The debug/trace system is not reset through this reset if debugger is connected. RTC module is not reset with WDR_N.

For more details refer to 4. Watchdog Timer.

2.3.3.3 Soft reset

Soft reset, resets core and peripheral domain. But mode, security and clock settings are unchanged. There is only one reset under this category and that is - software reset.

SWR_N (software reset)

Software reset can be achieved by writing 0x5A in SYSC_RSTCNTR:SWRST register. This reset facilitates the user to reset the software and start from BootROM execution again. Since the System Controller is not reset during this time, all profile settings (power and clock) are unchanged. Even debug and trace logic is unaffected by this reset.

2.3.3.4 Power domain reset

This reset is generated by the power control module before it has to power up the particular power domain. After power is switched ON for that domain, the domain is initialized to default value. This is done to ensure all flops are initialized before isolation is removed. For more details refer to 2.3.4 Power management. There are four resets corresponding to four switchable power domain - PD2R_N, PD3R_N, PD4R_N and PD5R_N.

PD2R_N (power domain - 2 reset)

This resets power domain 2 which contains all peripherals and Security Checker. For more details on power domains refer to 2.3.4 Power management. BootROM execution starts once device to come out of this reset. Since security information is lost with power domain 2 reset, security evaluation is required once again. To avoid any clock manipulation RUN profile (which needs to be applied at wakeup) needs to use RC clock as system clock. This is also ensured by profile checking in state controller.

PD3R_N (power domain - 3 reset)

This resets power domain 3 which contains core group. BootROM execution starts once device comes out of this reset, but security evaluation is not required. Clock settings are also unchanged.

PD4R_N (power domain - 4 reset)

This resets power domain 4 which contains Retention RAM. Mode, security, and clock settings are unchanged.

PD5R_N (power domain - 5 reset)

This resets power domain 5 which contains Graphics Subsystem. Mode, security, and clock settings are unchanged.

2.3.3.5 JTAG reset

This reset group has two reset source: nTRST, which resets TAP

- DBGR_N, debug and trace system, which resets only debug and trace module Mode, security, and clock settings are unchanged for these reset sources. nTRST

This reset is generated by the external pin nTRST which resets Debug/Trace system and TAP Controller asynchronously. nTRST is synchronized into the coresight system. No other logic is affected by this reset.

Note: Debug and trace system gets reset by the falling edge of nTRST. Later, the user has a chance to configure the debug and trace system internally when no debugger is connected.

As the debug and trace system is distributed in Power Domain 2 and 3, it is not possible to debug real power up behavior of power domain 2 or 3. Fake power down mode is introduced to enable this debugging. For more details refer to 2.3.4.5 Fake power down mode.

DBGR_N

This reset is generated by the System Controller when SYSC_RSTCNTR:DBGR bits are configured with 0xDA. This reset, resets the debug and trace system.

2.3.3.6 Effect of reset sources

Table 2-111 shows reset sources and its effect on the device.

Table 2-111. Overview of reset sources

Reset source	Components to get reset	Mode evaluation	Security evaluation	Clock after reset
Power reset (PRSTX)	Whole device except- TAP	Yes	Yes	CLK_RC
Pin reset (RSTX)	Whole device except following modules- TAP, Reset Cause Register, and debug/ trace module gets reset with this source of reset, if the debugger is not connected to the device.			
Software- triggered HW reset (SWHWR_N)				
Profile error reset (PRFERR_N)				
Watchdog reset (WDR_N)				
Clock Supervisor reset (CSVR_N)				
SW reset (SWR_N)	Whole device except following modules- TAP, security logic, debug/ trace system, watchdog configuration registers, and System Controller. Note: Watchdog counter gets reset due to software reset. For more details refer 4. Watchdog Timer	No	No	No change
Power domain 2 reset (PD2R_N)	All logic in power domain 2 (It includes peripherals, security logic and debug/trace system). Note: Reset to debug/trace system can be avoided by configurable SYSC_SPCCFGR:FAKEPWRCNT.	No	Yes	CLK_RC
Power domain 3 reset (PD3R_N)	All logic in power domain 3 (it includes core group, Flash, debug and trace logic in this domain, HSSPI etc.).		No	No change
Power domain 4 reset (PD4R_N)	All logic in power domain 4 (It includes Retention RAM).			
Power domain_N 5 reset (PD5R_N)	All logic in the power domain 5 (it includes Graphics Subsystem)			
JTAG reset (nTRST)	TAP and debug/trace system.	No	No	No change
Debug and trace reset (DBGR_N)	Debug/trace system.	No	No	No change

2.3.3.7 Reset, system clock and stabilization wait times

The device has many reset causes. Refer to the Table 2-111 above for the device clock status after each reset. The stabilization wait time depends on the reset cause and the status of the RC oscillator when the reset occurred.

2.3.3.8 Stabilization wait time in case of a hard reset

The stabilization wait time in case of a hard reset consists of two parts, the Hard reset extension time and the RC clock stabilization time. The hard reset extension time is required since at hard reset LVD setting are set to default value and it needs a stabilization time of 2343 RC clocks. Hard reset extension counter starts counting with the deassertion of hard resets (RSTX, PRSTX, SWHWR_N, PRFERR_N, WDR_N, CSVR_N), while the RC clock stabilization time starts counting after hard reset extension time is over. After reset extension counter expires, RC clock stabilization time of 1920 RC clock cycles is counted. The switchable power domains are switched ON in sequence with power domain 3 switched ON in the end. This will take 5156 RC clock. CPU starts executing the boot code with RC clock.

Stabilization of the Slow RC oscillator

The initial value of Slow RC oscillation stablilization wait time selector is set to 64 Slow RC clock cycles (SYSC_SRCSCTCPR:PSCL=0x6, SYSC_SRCSCTCPR:CMPR = 0x0001).

The Slow RC oscillation stabilization wait time is measured with the Slow RC clock timer which starts counting when hard reset extension time has expired.

Stabilization of the RC oscillator

The initial value of the RC oscillation stabilization wait time selector is set to 1920 RC clock cycles (SYSC_RCSCTCPR:PSCL = 0x6, SYSC_RCSCTCPR:CMPR = 0x001E).

The RC oscillation stabilization wait time is measured with the RC clock timer which starts counting when hard reset extension time has expired.

Stabilization of the Main oscillator

The initial value of the main oscillation stabilization wait time selector is set to 32 Slow RC clock cycles (SYSC_SRCSCTCPR:PSCL=0x5, SYSC_SRCSCTCPR:CMPR = 0x1000). For ATLAS and CALYPSO devices, the initial value of the Main oscillation stabilization wait time selector is set to 2^{18} Main clock cycles (SYSC_MAINSCTCPR:PSCL = 0x6, SYSC_MAINSCTCPR:CMPR = 0x1000) and Main oscillator is enabled (SYSC_APPCKSRER:MOSCEN = ’1’). This value can be overwritten directly after start of the user program execution (when the MCU is running on RC clock). Select a value suitable for the used oscillator.

The Main oscillation stabilization wait time is measured with the Main clock timer which starts counting when hard reset extension time has expired.

Stabilization of the Sub oscillator

The initial value of the Sub oscillation stabilization wait time selector is set to 2^{16} Sub clock cycles (SYSC_SUBSCTCPR:PSCL = 0x6, SYSC_SUBSCTCPR:CMPR = 0x400), and Sub oscillator is enabled (SYSC_APPCKSRER:SOSCEN = ’1’). This value can be overwritten directly after start of the user program execution (when the MCU is running on RC clock). Select a value suitable for the used oscillator.

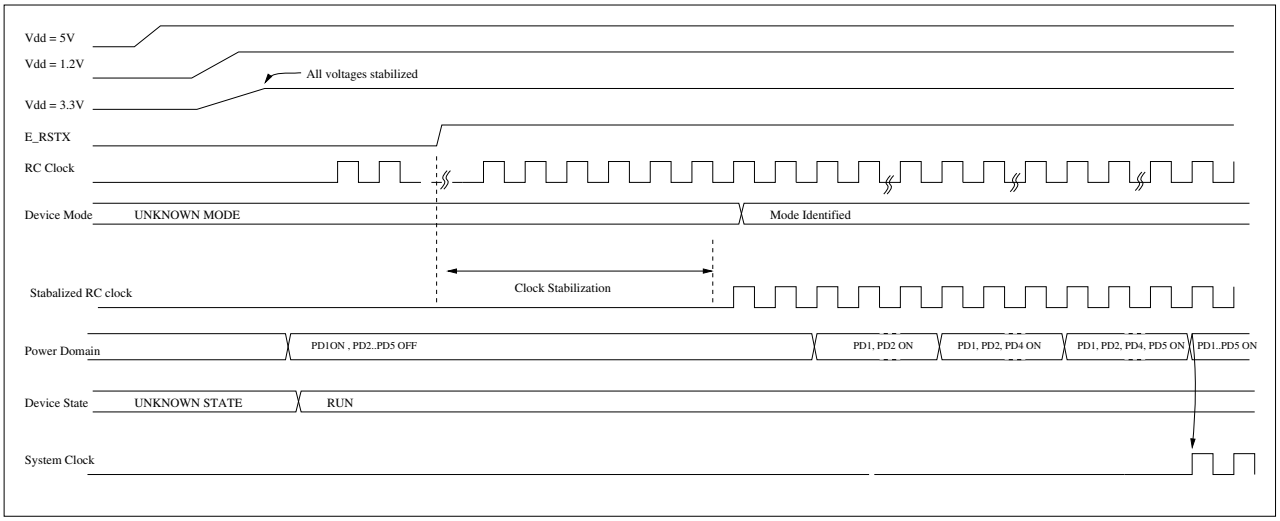
The Sub oscillation stabilization wait time is measured with the Sub clock counter which starts counting when hard reset extension time has expired.

2.3.3.9 Startup after power and external reset

After a power or external reset event, the system waits for the stabilization of the power supply and the RC oscillator. The switchable power domains are switched ON in sequence with power domain 3 switched ON in the end. Then the CPU starts executing the BootROM program with the RC clock as clock source.

A transition to another clock (Main clock, Sub clock, Slow RC clock, PLL clock, SSCG PLL clock, or GFX PLL clock) is possible after stabilization of the respective clock. The stabilization times for the external oscillators can be adjusted to the characteristics of the connected oscillators.

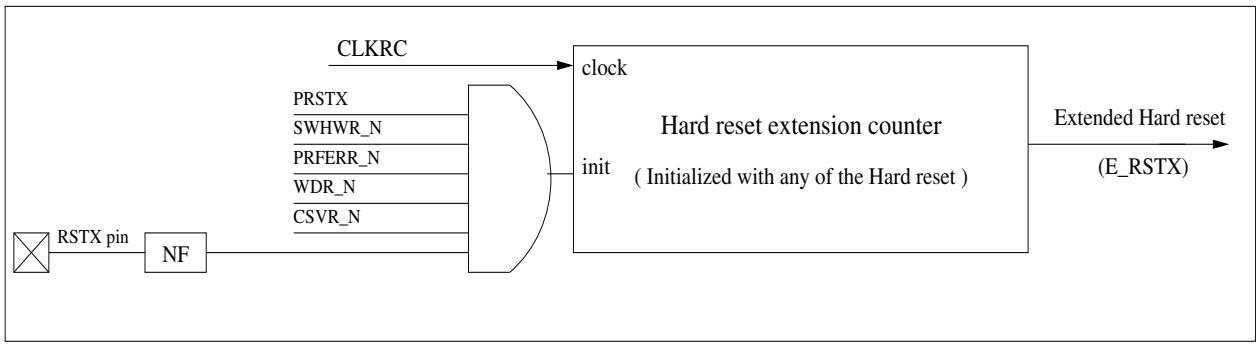
Figure 2-115. Device power up sequence



2.3.3.10 Reset extension

All hard reset events are extended by the Hard reset extension circuit to guarantee the stabilization of the Low Voltage Detector (LVD) and complete reset of the device before program execution starts.

Figure 2-116. Block diagram of reset extension circuit



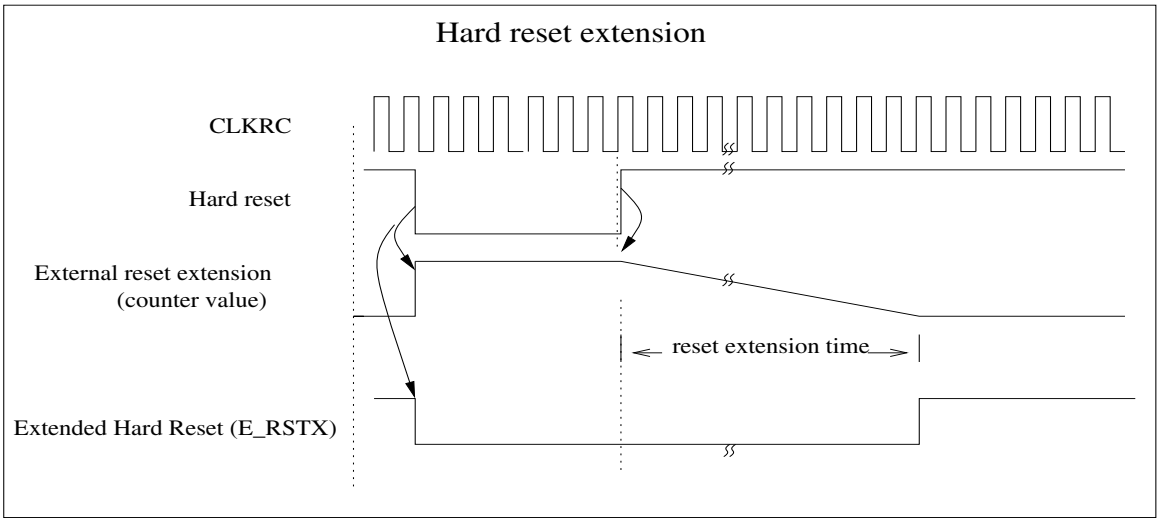
Hard reset extension

The Hard reset extension counter is initialized by Hard reset (PRSTX, SWHWR_N, PRFERR_N, WDR_N, CSVR_N & RSTX) and is clocked by the RC clock.

This counter keeps the extended Hard reset (E_RSTX) active for 2343 RC clock cycles after the last hard reset is deasserted. This is required for LVD stabilization time.

Figure 2-116 shows a block diagram of the reset extension circuit and Figure 2-117 shows functional timing diagram of external reset extension function.

Figure 2-117. Function of the hard reset extension circuit



2.3.3.11 Source clock stabilization

The Source Clock Timers (RC clock timer, Slow RC clock timer, Main clock timer, and Sub clock timer) and the clock monitor bits (SYSC_CKSRESTSR:RCCLKRDY, SYSC_CKSRESTSR:SRCCLKRDY, SYSC_CKSRESTSR:MAINCLKRDY, SYSC_CKSRESTSR:SUBCLKRDY, SYSC_CKSRESTSR:MAINPLLRDY, SYSC_CKSRESTSR:SSCGPLLRDY and SYSC_CKSRESTSR:GFXPLLRDY) are all cleared by all hard reset event even while the four oscillators are enabled.

RC clock timer, Slow RC clock timer, Main clock timer, Sub clock timer

All the Source Clock Timers are cleared by E_RSTX. They start counting the stabilization time when E_RSTX is released. During the stabilization time the respective clocks are gated and not available.

2.3.3.12 Start of program execution after hard reset

The RC clock timer starts counting after deactivation of E_RSTX. After 1920 RC clock cycles, the RC clock ready monitor bit SYSC_CKSRESTSR:RCCLKRDY will be set and the RC clocks are available. All switchable power domains are switched ON in sequence with power domain 3 (CPU) switched ON in the end. After power domain release, the CPU is always in RC clock mode with the RC clock frequency set to nominal 12 MHz and starts executing the BootROM program. The Low Voltage Detector level (SYSC_APPLVDCFGR:SVx) is initialized to the lowest possible value. If a higher detection value is needed, the register value can be changed accordingly.

For more details refer to Figure 2-117.

2.3.3.13 Power domain status

After reset extension, all switchable power domains are fully switched ON in sequence. For details refer to Figure 2-115.

2.3.3.14 Operation of the low-voltage reset function

This section describes the operation of the low-voltage reset function.

Function of the low-voltage reset

The low-voltage reset function uses the Low-Voltage Detector that compares the power supply voltage VCC with an internally generated reference voltage. This reference voltage is programmable with the SV50/SV33/SV12 bits of SYSC_RUNLVDCFGR and SYSC_PSSLVDCFGR registers.

If the Low-Voltage Detector for 1.2 V voltage supply is enabled (SYSC_RUNLVDCFGR:LVDE12/ SYSC_PSSLVDCFGR:LVDE12 is set), the Low-Voltage Detector for 5.0 V voltage supply is enabled (SYSC_RUNLVDCFGR:LVDE50/ SYSC_PSSLVDCFGR:LVDE50 is set), the Low Voltage Detector for 3.3 V voltage supply is enabled (SYSC_RUNLVD-

CFGR:LVDE33/ SYSC_PSSLVDCFGR:LVDE33 is set) and the corresponding Low Voltage Detector reset is enabled (LVDR12/LVDR33/LVDR50 bits of SYSC_RUNLVDCFGR/SYSC_PSSLVDCFGR register is set), and the voltage supply falls below the selected detection level SV12 (select voltage level for 1.2 V supply), SV33 (select voltage level for 3.3 V supply) or SV50 (select voltage level for 5.0 V supply), then the reset extension counter is initialized and the PRSTX flag is set. See the datasheet for the specification of the selectable detection levels SV12, SV33, and SV50.

After recovery of the power supply voltage, the hard reset extension counter is released and the startup is performed as described in 2.3.3.9 Startup after power and external reset.

Configuration of the Low-Voltage Detector

Stabilization time requirement

The Low-Voltage Detector (LVD) needs stabilization time of 110 µs, which is applied each time LVD is enabled or detection level setting is changed. At each hard reset LVD is enabled, detection level settings are reset to 'Level 0' and its stabilization time is taken care by hard reset extension counter. During operation if LVD detection level setting is changed or is disabled and enabled again then stabilization counter starts and LVD output is masked till stabilization counter expires. LVD stabilization time can be checked by LVD ready bit (LVDRDY12, LVDRDY33 and LVDRDY50) in SYSC_LVDCFGSTSR register. LVD RDY bit stays high when LVD is functional, it goes low as soon as LVD setting is changed and goes high once again when LVD is enabled and stabilization counter expires.

Low-Voltage Detector

The LVD 1.2 V reset is controlled by the LVD 1.2 V Enable (SYSC_RUNLVDCFGR:LVDE12/ SYSC_PSSLVDCFGR:LVDE12), LVD 1.2 V Reset Enable (SYSC_RUNLVDCFGR:LVDR12/SYSC_PSSLVDCFGR:LVDR12), and LVD detection level selection (SYSC_RUNLVDCFGR:SV12/ SYSC_PSSLVDCFGR:SV12) bits. Setting the SYSC_RUNLVDCFGR:LVDE12/SYSC_PSSLVDCFGR:LVDE12 bit to '0' disables the Low Voltage Detector and setting the bit to '1' enables the detector. Setting the SYSC_RUNLVDCFGR:LVDR12/SYSC_PSSLVDCFGR:LVDR12 bit to '0' disables the LVD reset and provides interrupt if 1.2 V voltage supply goes low compared to configured detection level. Setting the SYSC_RUNLVDCFGR:LVDR12/ SYSC_PSSLVDCFGR:LVDR12 bit to '1' (which is the default value after any hard reset) enables the LVD reset. If 1.2 V voltage supply goes lower than configured detection level then it generates LVD reset. After each hard reset, SYSC_RUNLVDCFGR:LVDE12/SYSC_PSSLVDCFGR:LVDE12 bit is set to '1' (LVD 1.2 V enabled), SYSC_RUNLVDCFGR:LVDR12/SYSC_PSSLVDCFGR:LVDR12 bit is set to '1' (LVD will generate reset on low voltage detection) and SYSC_RUNLVDCFGR:SV12 level select bits are reset to '000' (Level 0) by any reset.

If LVD is enabled, LVD reset is disabled (SYSC_RUNLVDCFGR:LVDR12/ SYSC_PSSLVDCFGR:LVDR12 bit is '0') then any Low Voltage Detection at 1.2 V voltage supply will set the interrupt flag (SYSC_SYSERRR:LVD12IF) and interrupt will be generated. This interrupt can be cleared by writing '1' to the SYSC_SYSERRICLR:LVD12ICLR bit.

The detection level is always set to 'Level 0' after each hard reset. This level can be increased after startup by setting the LVD configuration through profile registers. For more details refer to 2.3.2.4 RUN to RUN state switching.

The LVD for 5 V[a] and 3.3 V are similar as Low-Voltage Detector 1.2 V.

Effect on current consumption

The LVD draws a current when it is activated (for details refer to datasheet). This current flows independent of the selected operation mode. If this is not acceptable in PSS, user can disable the LVD by applying the new configuration through PSS profile register.

When RC oscillator is enabled in the profile, the 5 V Low-Voltage Detector macro is enabled. This is done as the bandgap functionality of LVD is used by RC oscillator for reference voltage.

Note: The low-voltage reset function will not be available in this case.

During wakeup the LVD must be enabled, and this needs to be taken care in the RUN profile before going to PSS. Otherwise validity check fails, profile error NMI is generated, and device will not enter into PSS state.

a. While Flash programming or Flash erase operation is ON and LVD reset on 5 V is enabled (SYSC_APPLVDCFGR:LVDR50 bit is set to '1' and SYSC_APPLVDCFGR:LVDE50 bit is set to '1') then do not set LVD reference voltage selection bit (SYSC_APPLVDCFGR:SV50) below 2.7 V. The reference voltage in this case can be calculated based on the below parameter:
   - Flash tready time

- LVD precision
- LVD detection time
- Minimum flash operating voltage
- Current consumption and capacitance

For the above parameters refer to the Flash and LVD datasheet.

2.3.4 Power management

2.3.4.1 Overview

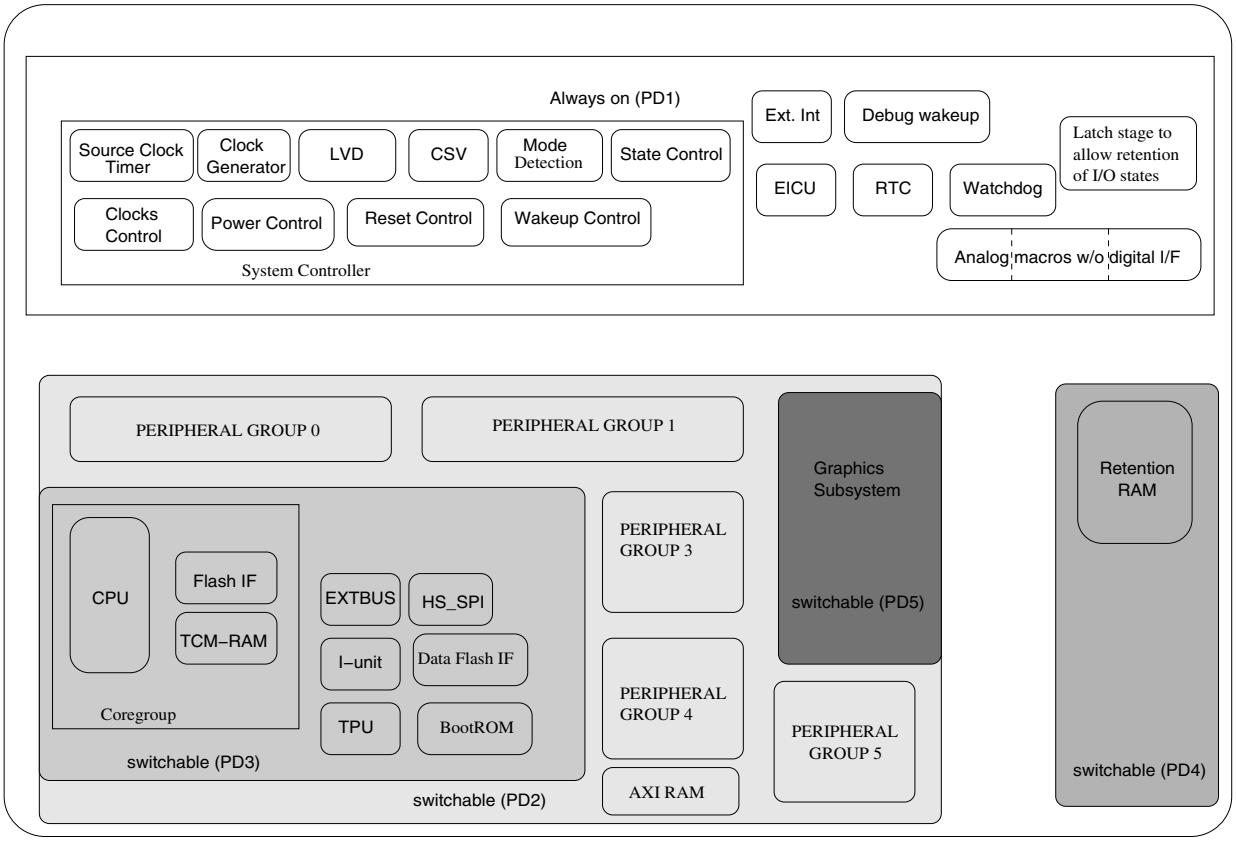
There are five power domains defined for the device. They are as below:

- Always ON power domain (PD1)
- Peripheral power domain (PD2)
- Core power domain (PD3)
- Retention RAM power domain (PD4)
- Graphics Subsystem power domain (PD5)

Always ON domain contains device control logic i.e. clock, reset and power control, RTC for timer support, and Retention RAM for device state saving. PD2, PD3, PD4, and PD5 are switchable domains, which can be switched OFF by software to save leakage power. Power domains, PD3 and PD5 are nested in the power domain in PD2, which means PD2 needs to be ON, if PD3 or PD5 is ON.

Figure 2-118 shows device power partitioning.

Figure 2-118. Device power partitioning



2.3.4.2 Power domain control

Device has four switchable domains which are controlled by the power domain configuration register defined in the RUN profile i.e. SYSC_RUNPDCFGR and PSS profile i.e. SYSC_PSSPDCFGR. The power control module, based on the profile settings, switches ON or switches OFF the power domains, following switch ON and switch OFF sequence as defined in 2.3.4.3 Power control mode.

All the status of the power domains i.e. registers and RAMs are lost once corresponding domain is powered down. The software needs to re-configure the domain, after power up of the domain.

The state of the power domain i.e. whether domain is switched ON or OFF is reflected in the Device Status Register SYSC_PDSTSR:PDxSTS. The data or instruction access to the switched OFF power domain generates data or instruction abort.

The typical switch ON of a power domain takes around 60 μs.

Note: Because of nested power domain structure in the device, PD2 can never be turned OFF, if PD3 or PD5 is ON. These combinations are rejected by the device state logic and error status is set.

For more details on how the profile settings are applied refer to 2.3.2.3 Device state switching.

2.3.4.3 Power control mode

The power controller supports two modes of operation:

- Sequential power domain control
- Fast power domain control

The mode can be configured by the configuration register SYSC_SPCCFGR:FASTON

Sequential power domain control

The power controller works in this mode, when configuration register SYSC_SPCCFGR:FASTON is set to ’0’. This is the default mode of power controller. In this mode power switch ON requests are processed in sequence. The switch ON request is processed in below priority:

- PD2 is switched ON
- PD4 is switched ON
- PD5 is switched ON
- PD3 is switched ON

The switch OFF requests are processed in parallel, after switch ON is done.

This mode of operation reduces the voltage drop of the chip introduced due to power switch noise. At the same time the wake up time of the device increases.

On reset, the device always switches on the power domain in sequence. This avoids voltage drop and hence the loss of RAM contents.

Fast power domain control

The power controller works in this mode, when configuration register SYSC_SPCCFGR:FASTON is set to ’1’.

In this mode all power switch ON/ OFF requests are processed in parallel. This reduces the wake up time of the system. But at the same time it can cause a bigger step for the change of the supply current. For the current consumption of the device during switching ON the power domains, refer to the datasheet. Check the regulator capability to avoid voltage drop below the minimum required supply voltage.

Figure 2-119 and Figure 2-120 show power domain control in both modes.

Figure 2-119. Power ON switching with SYSC_SPCCFGR:FASTON = ‘1’

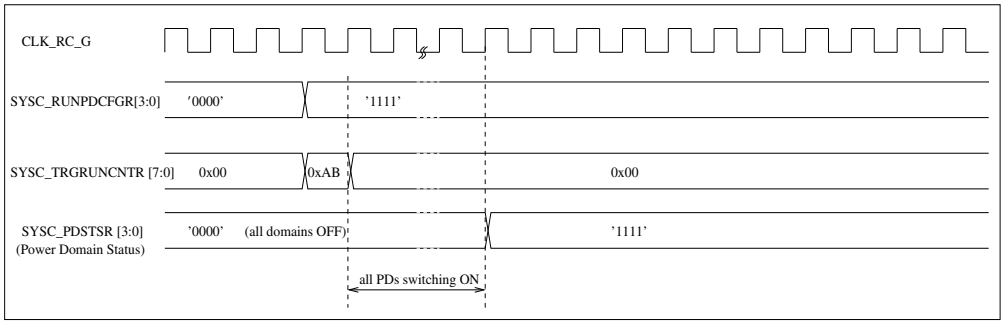
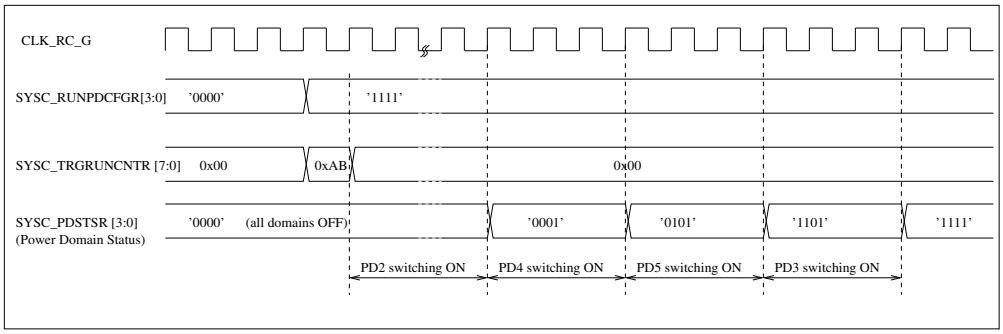


Figure 2-120. Power ON switching with SYSC_SPCCFGR:FASTON = ‘0’

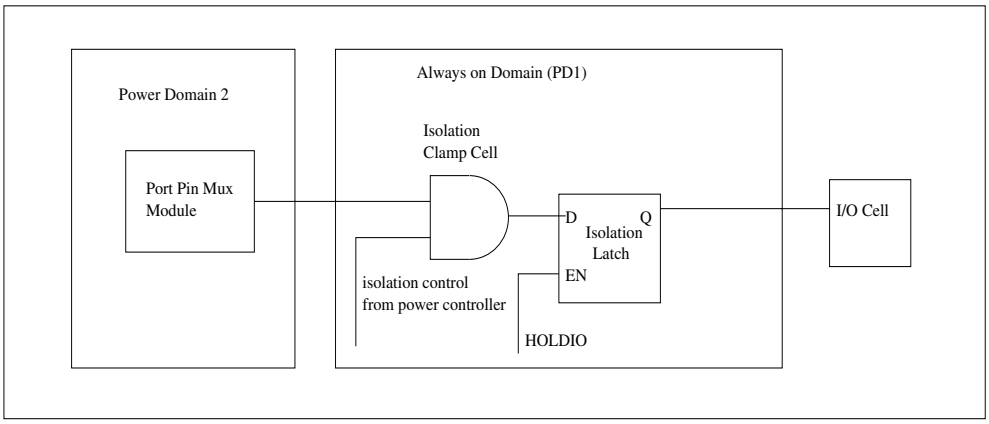


2.3.4.4 IO data retention at power OFF

When a certain power domain on the device is powered down, then all the output signals which goes to a possibly powered-on region are held at a stable value (either VDD or GND). This prevents the subsequent logic which are still powered ON to get disturbed through unstable, unpredictable logic values of the outputs of the power domain which is switched OFF. This is ensured by the power control module. It may be required that when the device power domains are switched OFF, the I/Os of the device should not change (continue to hold the previous state), which otherwise may cause erroneous operation in other ICs connected to this device. For this requirement, device implements retention latch strategy for the I/Os, so that when the core power is turned OFF, the value driven on I/Os are latched with previous data. These isolation latches are controlled by the software.

The isolation cell structure and connection are shown in Figure 2-121.

Figure 2-121. IO data retention at power OFF



The sequence for switching OFF power domain 2 is as follows:

1. Write SYSC_SPCCFGR:HOLDIO = '1'. The current values driven on IO are latched in isolation latch.
2. Save the Pin Configuration Registers to Retention RAM or System RAM.
3. Configure the required power domain configuration in profile register and trigger the profile for application.
4. Now power controller will activate the isolation and switch OFF the power domain.

Note: User needs to ensure that peripheral output does not toggle, once the peripheral power domain switch off has begun.

The sequence for switching on power domain 2 is as follows:

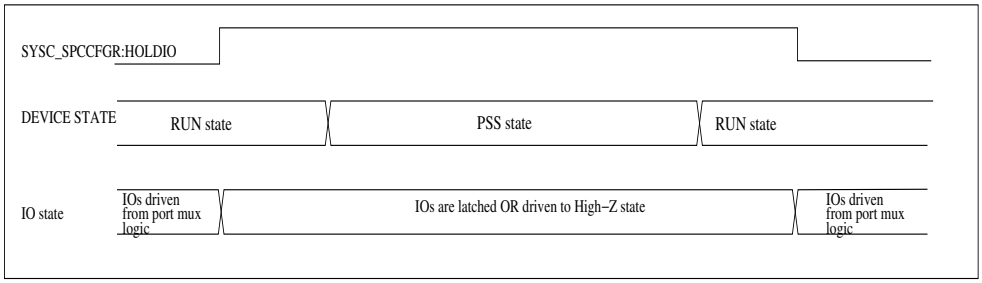
1. Wake up event triggers device state logic, which in turn triggers power controller.
2. Power controller will assert the reset, switch ON the power domain, and deactivate the isolation cell.
3. Power controller will deassert the reset and start the clock.
4. Software needs to restore the configuration of the power domain from Retention or System RAM.
5. Write SYSC_SPCCFGR:HOLDIO = '0'. This opens up isolation latch.

Note: User needs to ensure the stability of the Port Pin Mux module output, between step 4 and step 5.

During SYSC_SPCCFGR:HOLDIO signal assertion and subsequent PSS state entry, the IOs can continue to be in same state as they were in RUN state. They can also be driven by common control signals to a state for further power reduction depending on SYSC_SPCCFGR:PSSPADCTRL bit settings. For more details on these settings refer to 2.2.16.1 Special Configuration Register (SYSC_SPCCFGR).

Figure 2-122 shows the timing diagram of IO state change during device state switching.

Figure 2-122. IO state change during state switching



Note: If the SYSC_SPCCFGR:PSSPADCTRL bit is set to ’0’, then all output pad retain their previous value and input buffers will retain their previous enable or disable status. For more details refer to 2.2.14.1.

2.3.4.5 Fake power down mode

Since debug and trace logic is present in power domain 2 (PD2) and power domain 3 (PD3), it is not possible to debug actual power down and power up behavior of PD2 and PD3. Fake power down mode is introduced to simulate this scenario. During this mode even if PSS profile is configured to power down PD2 and PD3, power switches are kept ON so that debug and trace logic does not lose its configuration, but other logic in these domains are kept in reset to emulate power down of PD2 and PD3. Isolation cells are not active during the fake power down, which ensures that debugger does not lose connection to the device.

The fake power domain can be activated by configuring SYSC_SPCCFGR:FAKEPWRCNT bit to ’1’.

Note:

From application point of view, there is no functional difference of fake power down mode and normal power down. Only additional leakage current of PD2 and PD3 is drawn. This mode is not recommended to be used during the normal application scenario.

If debugger is connected, one needs to set SYSC_SPCCFGR:FAKEPWRCNT mode before powering down the PD2/PD3 otherwise debugger will loose connection and all configurations. If SYSC_SPCCFGR:FAKEPWRCNT is not set then it needs to pull the nTRST low during power- down and try a fresh connection with all configuration once device is up.

2.3.5 Clock management

Clock management of entire device is performed by clock control module. It is comprised of two major building blocks:

- Clock generation block which provides all source clocks
- Clock selection and distribution block which provides clock to each IP depending on the clock profile selected by the user

2.3.5.1 Clock overview

There are four different clock sources and three PLLs are available on this device. All PLLs use Main oscillator as input clock.

Source of clocks

- Main oscillator (typ. 4 MHz)
- Sub oscillator (typ. 32.768 kHz)
- RC oscillator (typ. 12 MHz)
- Slow RC oscillator (typ. 100 kHz)
- Main PLL (non-modulated clock, refer to device specific datasheet for maximum Main PLL output frequency)
- SSCG PLL (modulated clock (optional), refer to device specific datasheet for maximum Main PLL output frequency)
- GFX PLL (modulated clock (optional), refer to device specific datasheet for maximum Main PLL output frequency)

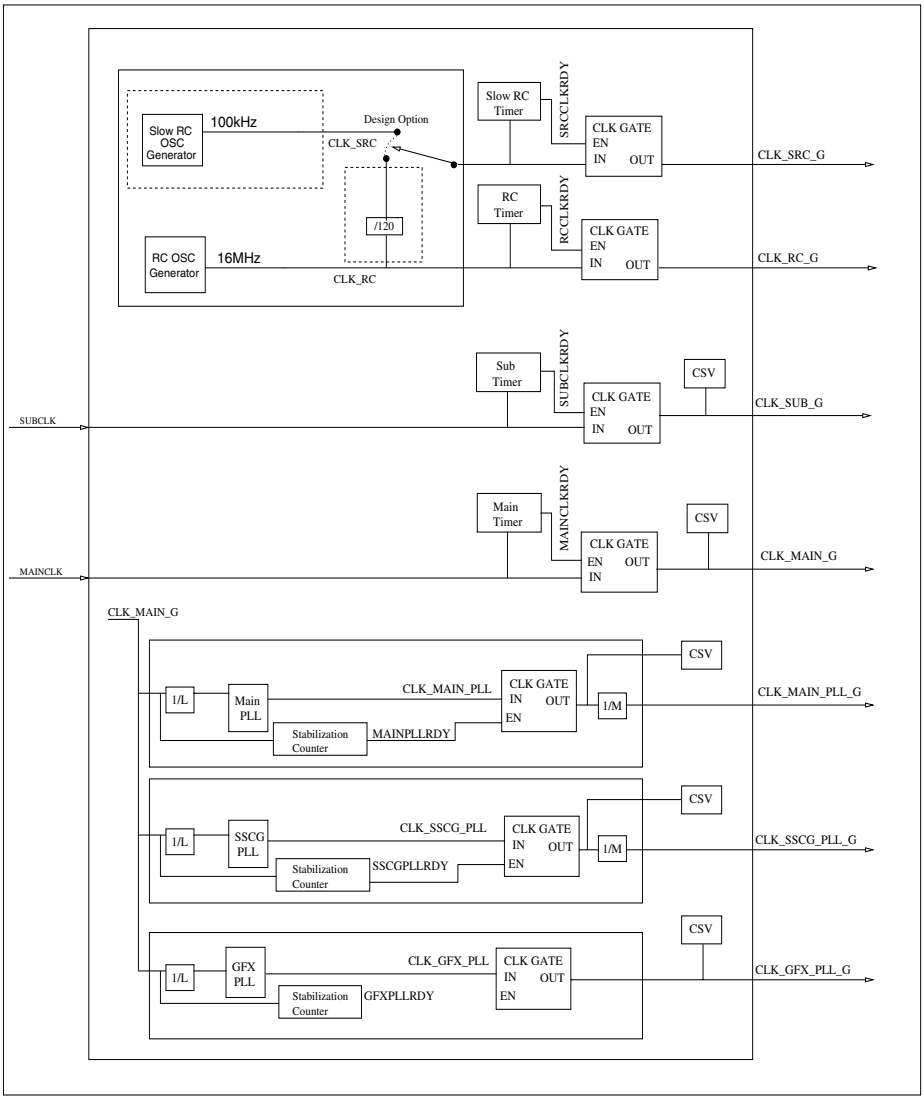
Features

- All four clock sources have Source Clock Timer, which mask the clock till it is stable

- All PLLs have stabilization counters, which masks the clock till it is stable
- All Source Clock Timers can also be used as standard timers after stabilization time is over
- Clock Supervisor circuits are present for both external clock sources (Main osc & Sub osc) which make sure that input clocks are within configured frequency ranges
- Clock Supervisor circuits are also present for all PLLs to detect if PLL goes out of lock due to high jitter on the input clock
- RC clock is used for complete device during initial power up, user can switch to other clocks later
- Slow RC oscillator is provided for low power timer modes since its current consumption is significantly lower compared to RC oscillator
- Main PLL provides non-modulated clock to required IPs, which cannot use modulated clock
- SSCG PLL (SSCG = Spread-Spectrum Clock Generation) provides modulated clock for EMI reduction
- GFX PLL provides modulated clock for Graphic Subsystem for pixel operation
- Glitch free clock muxes support dynamic clock switching
- Only even clock division ratios are supported
- Support for power saving using clock gating feature

2.3.5.2 Clock generation block

Figure 2-123. Clock generation block diagram



Note: Clock Supervisor (CSV) module for PLL is kept before divider-M to make the CSV configuration independent of clock division value. All CSVs (including PLL CSV) are disabled till stabilization time is over. This avoids any CSV reset during clock stabilization phase.

2.3.5.3 Source clocks

Main clock

The Main clock (MAINCLK) is provided by the Main oscillator which is connected externally to a quartz crystal or resonator. Alternatively an externally generated clock can be connected to the Main oscillator. The typical input frequency for Main oscillator is 4 MHz.

Dedicated Source Clock Timer is used to check the stabilization time for Main oscillator. If SYSC_CKSRESTSR:MAINCLKRDY register bit is set, Main clock is ready for use. For more details refer to 2.3.6 Source Clock Timers.

Dedicated Clock Supervisor (CSV) is used to supervise the frequency of Main oscillator. If corresponding CSV is enabled (for example, SYSC_RUNCSVCFGR:MOCSVE is set during RUN state) and clock is used in the device, any deviation of clock from configured range causes device reset. For more details refer to 2.3.7 Clock Supervisor.

Sub clock

The Sub clock (SUBCLK) is provided by the Sub oscillator which is connected externally to a quartz crystal or resonator. Alternatively an externally generated clock can be connected to the Sub oscillator. The nominal frequency is 32.768 kHz.

Dedicated Source Clock Timer is used to check the stabilization time for Sub oscillator. If SYSC_CKSRESTSR:SUBCLKRDY register bit is set, Sub clock is ready for use. Refer to 2.3.6 Source Clock Timers.

Dedicated Clock Supervisor (CSV) is used to supervise the frequency of Sub oscillator. If corresponding CSV is enabled (for example, SYSC_RUNCSVCFGR:SOCSVE is set during RUN state) and clock is used in the device, any deviation of clock from configured range causes device reset. For more details refer to 2.3.7 Clock Supervisor.

RC clock

The RC clock (CLK_RC) is provided by the RC oscillator internal to device. The resonance frequency shall be 12 MHz. For more details refer to the datasheet.

Dedicated Source Clock Timer is used to check the stabilization time for RC oscillator. If SYSC_CKSRESTSR:RCCLKRDY register bit is set, RC clock is ready for use. For more details refer to 2.3.6 Source Clock Timers.

Slow RC clock

The Slow RC (CLK_SRC) is provided by the Slow RC oscillator internal to device. The resonance frequency shall be 100 kHz. For more details refer to the datasheet.

Dedicated Source Clock Timer is used to check the stabilization time for Slow RC oscillator. If SYSC_CKSRESTSR:SRC-CLKRDY register bit is set, Slow RC clock is ready for use. For more details refer to 2.3.6 Source Clock Timers.

Main PLL clock

The Main PLL clock (CLK_MAIN_PLL) provides a high frequency unmodulated clock for the MCU which is derived from the Main clock (MAINCLK). Refer to device specific datasheet for Main PLL output frequency range. The PLL multiplication factor should be programmed accordingly.

Note:

If maximum supported frequency of the device is 160 MHz, user should use the divider-M setting to avoid CLK_MAIN_PLL frequency more than 160 MHz.

Configurable stabilization counter is used to check the stabilization time for Main PLL. If SYSC_CKSRESTSR:MAINPLLRDY register bit is set, Main PLL clock is ready for use. For more details refer to 2.3.5.5 PLL interface.

Dedicated Clock Supervisor (CSV) is used to supervise the frequency of Main PLL. If corresponding CSV is enabled (for example, SYSC_RUNCSVCFGR:MPSCVE is set during RUN state) and clock is used in the device, any deviation of clock from configured range causes device reset. For more details refer to 2.3.7 Clock Supervisor.

SSCG PLL clock

The SSCG PLL clock (CLK_SSCG_PLL) provides a high frequency modulated clock for the MCU which is derived from the Main clock (MAINCLK). Refer to device specific datasheet for SSCG PLL output frequency range. The PLL multiplication factor should be programmed accordingly.

Note:

If maximum supported frequency of the device is 160 MHz, user should use the divider-M setting to avoid CLK_SSCG_PLL frequency more than 160 MHz.

It supports programmable modulation rate (0.5%.. 5%) and modulation frequency (Fin/1024, Fin/ 2048, Fin/4096). It supports two types of modulation - center spread and down spread type modulation. Modulation can also be disabled if very precise clock is required.

Configurable stabilization counter is used to check the stabilization time for SSCG PLL. If SYSC_CKSRESTSR:SSCGPLL-RDY register bit is set, SSCG PLL clock is ready for use. For more details refer to 2.3.5.5 PLL interface.

Dedicated Clock Supervisor (CSV) is used to supervise the frequency of SSCG PLL. If corresponding CSV is enabled (for example, SYSC_RUNCSVCFGR:SPSCVE is set during RUN state) and clock is used in the device, any deviation of clock from configured range causes device reset. For more details refer to 2.3.7 Clock Supervisor.

GFX PLL clock

The GFX PLL clock (CLK_GFX_PLL) provides a high frequency clock with optional/configurable modulated for Graphics Sub-system which is derived from the Main clock (MAINCLK). Refer to device specific datasheet for SSCG PLL output frequency range. The PLL multiplication factor should be programmed accordingly.

It supports programmable modulation rate (0.5%.. 5%) and modulation frequency (Fin/1024, Fin/ 2048, Fin/4096). It supports two types of modulation - center spread and down spread type modulation. Modulation can also be disabled if very precise clock is required.

Configurable stabilization counter is used to check the stabilization time for GFX PLL. If

SYSC_CKSRESTSR:GFXPLLRDY register bit is set, GFX PLL clock is ready for use. For more details refer to 2.3.5.5 PLL interface.

Dedicated Clock Supervisor (CSV) is used to supervise the frequency of GFX PLL. If corresponding CSV is enabled (for example, SYSC_RUNCSVCFGR:GPCSVE is set during RUN state) and clock is used in the device, any deviation of clock from configured range causes device reset. For more details refer to 2.3.7 Clock Supervisor.

2.3.5.4 Source clock usage

Enabling and disabling of source clocks

- After each hard reset, the Main oscillator, Sub oscillator and RC oscillator are enabled while the, Slow RC oscillator and all PLL multiplier circuits are disabled
- Source clocks not used in the device can be enabled and disabled in the profile with the corresponding enable bits in the profile
- Enable a source clock if it is needed for a certain function or for a fast system clock changing by setting the corresponding enable bit to ’1’. After stabilization of the activated clock, the corresponding clock monitor bit is set and indicates the clock as ‘ready’
- Source clock cannot be disabled if it is used in the device, this is checked by profile validity checks
- To go to deep power save state, switching all clock muxes to ‘no-clock’ and switching OFF all clock sources can be done together in the PSS profile
- The Main clock must be enabled if the PLL clock should be enabled, otherwise profile error interrupt will be asserted

Stabilization time requirement

When the power is switched ON, when hard reset is released or when a disabled oscillator is enabled, a stabilization wait time is required before the clock can be used.

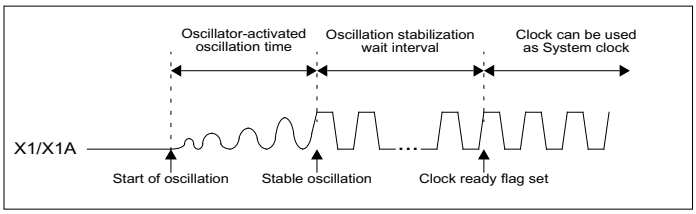
Ceramic and crystal oscillators which can be connected to the X0/X1 and X0A/X1A pins generally require several ms to stabilize at their natural frequency (oscillation frequency) when oscillation starts. The internal PLL multiplier circuits and the RC oscillators also need a certain stabilization time after activation. For this reason, device operation with the activated clock is not allowed immediately after oscillator is enabled but is allowed only after stabilization time.

When the clock source is switched, the MCU runs with the previously selected clock until newly selected clocks are stabilized. When the corresponding clock ready flag is set, the MCU changes to the specified clock.

All hard reset clears all Source Clock Timers and clock ready monitor bits and the corresponding oscillation stabilization wait interval is applied.

Figure 2-124 shows the operation of the oscillators directly after activation.

Figure 2-124. Operation immediately after oscillation starts

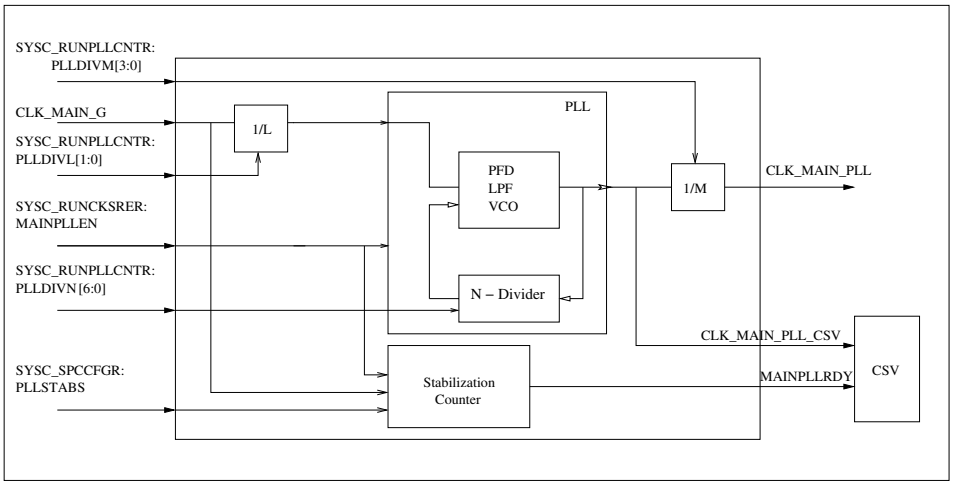


2.3.5.5 PLL interface

Main PLL interface

The PLL multiplier circuit is used to generate the PLL clock out of the Main clock.

Figure 2-125. PLL interface block diagram



Frequency calculation

$$\mathbf{CLK_MAIN_PLL} = \frac{\mathbf{CLK_MAIN_G} \times \mathbf{(SYSC_RUNPLLCNTR:PLLDIVN\ +1)}}{\mathbf{(divider_M)} \times \mathbf{(\ divider_L)}}$$

Where,

SYSC_RUNPLLCNTR:PLLDIVM=0 --> divider_M = 1

SYSC_RUNPLLCNTR:PLLDIVM>0 --> divider_M = (SYSC_RUNPLLCNTR:PLLDIVM * 2)

SYSC_RUNPLLCNTR:PLLDIVL=0 --> divider_L = 1

SYSC_RUNPLLCNTR:PLLDIVL>0 --> divider_L = (SYSC_RUNPLLCNTR:PLLDIVL * 2)

Table 2-112. Operating constraints for the Main PLL

Parameter	Symbol	Min	Typ	Max	Unit
Internal PLL input frequency (after divider L)	f_in	4	-	16	MHz
Internal PLL VCO frequency	f_vco	200	-	400	MHz

Table 2-112. Operating constraints for the Main PLL

Parameter	Symbol	Min	Typ	Max	Unit
Multiplication	N	13	-	100	-

Caution: Any configuration register (SYSC_RUNPLLCNTR:PLLDIVN, SYSC_RUNPLLCNTR:PLLDIVL) which effect PLL locking must not be changed while PLL is ON (SYSC_RUNCKSRER:MAINPLLEN is '1'), it is checked in the profile checker. But the SYSC_RUNPLLCNTR:PLLDIVM divider setting can be changed at any time. The same is true for the PSS profile.

Steps in programming the PLL interface

This section gives the general steps a programmer must follow while using the PLL interface. Following steps are required to enable PLL:

1. Update profile register for Main PLL (SYSC_RUNCKSRER:MAINPLLEN, SYSC_RUNPLLCNTR:PLLDIVL, SYSC_RUN-PLLCNTR:PLLDIVN, SYSC_RUNPLLCNTR:PLLDIVM or corresponding PSS profile register. Update profile register to select PLL clock (SYSC_RUNCKSELR or SYSC_PSSCKSELR). Update the profile register to select the PLL stabilization time (SYSC_SPCCFGR:PLLSTABS).
2. Set Profile Trigger Register or execute WFI instruction with SYSC_PSSENR:PSSEN = 0xBA.
3. Device state control logic first enables the PLL with configured settings, waits for the PLL stabilization time and then switches the clock source domain to PLL source.

To change the clock to some other clock:

1. Update the profile register to switch the clock source from PLL to other clock source. Update the profile register to switch OFF PLL (SYSC_RUNCKSRER:MAINPLLEN = '0' or SYSC_PSSCKSRER:MAINPLLEN = '0').

Note: For fast profile switching (for a fast wakeup forexample), unused slow source clocks, such as Slow RC, Osc and SUB Osc, should be disabled.

2. Set Profile Trigger Register or execute WFI instruction with SYSC_PSSENR:PSSEN = 0xBA.
3. Device state control logic will make sure that PLL switches OFF only once clock switching is complete.

SSCG and GFX PLL interface

The SSCG and GFX PLL multiplier circuit is used to generate the SSCG PLL clock out of the Main clock.

Figure 2-126. SSCG-PLL interface block diagram

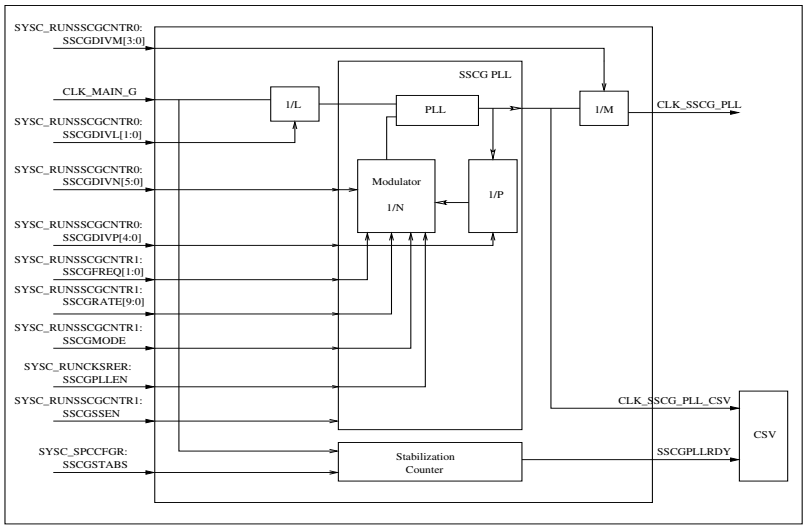
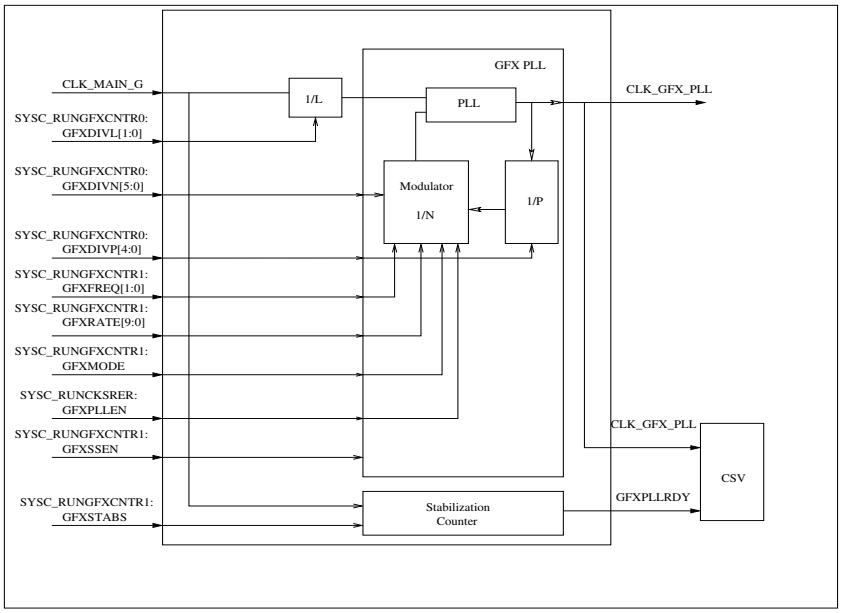


Figure 2-127. GFX PLL interface block diagram



Frequency calculation

$$CLK_SSCG_PLL = \frac{CLK_MAIN_G \times (SYSC_RUNSSCGCNTR0{:}\ SSCGDIVN) \times (SYSC_RUNSSCGCNTR0{:}SSCGDIVP)}{(SSCG_divider_M) \times (SSCG_divider_L)}$$

$$CLK_GFX_PLL = \frac{CLK_MAIN_G \times (SYSC_RUNGFXCNTR0{:}\ GFXDIVN) \times (SYSC_RUNGFXCNTR0{:}GFXDIVP)}{(GFX_divider_L)}$$

Where,

SYSC_RUNSSCGCNTR0:SSCGDIVM=0 --> SSCG_divider_M = 1

SYSC_RUNSSCGCNTR0:SSCGDIVM>0--> SSCG_divider_M = (SYSC_RUNSSCGCNTR0:SSCGDIVM * 2) SYSC_RUNSSCGCNTR0:SSCGDIVL=0 --> SSCG_divider_L = 1

SYSC_RUNSSCGCNTR0:SSCGDIVL>0--> SSCG_divider_L = (SYSC_RUNSSCGCNTR0:SSCGDIVL * 2) SYSC_RUNGFXCNTR0:GFXDIVL=0 --> GFX_divider_L = 1

SYSC_RUNGFXCNTR0:GFXDIVL>0 --> GFX_divider_L = (SYSC_RUNGFXCNTR0:GFXDIVL * 2)

Caution:

Any configuration register (SYSC_RUNSSCGCNTR0:SSCGDIVL, SYSC_RUNSSCGCNTR0:SSCGDIVN, SYSC_RUNSSCGCNTR0:SSCGDIVP, SYSC_RUNSSCGCNTR1:SSCGSSEN, SYSC_RUNSSCGCNTR1:SSCGMODE, SYSC_RUNSSCGCNTR1:SSCGFREQ, SYSC_RUNSSCGCNTR1:SSCGRATE) which effect SSCG PLL locking must not be changed while SSCG PLL is ON (SYSC_RUNCKSRER:SSCGPLLEN is '1'), it is checked in profile checker. But SYSC_RUNSSCGCNTR0:SSCGDIVM divider setting can be changed at any time.

Same is true for PSS profile.

Clock modulation

SSCG PLL supports two types of modulation mode:

1. Center spread clock modulation - clock is modulated keeping target frequency at center.
2. Down spread clock modulation - clock is modulated below target frequency.

Note: Since during center spread actual frequency can be more than configured one, it is recommended to use down spread during high speed (closer to max specified frequency) operation.

Figure 2-128 and Figure 2-129 show the effect of clock modulation on clock.

Figure 2-128. Clock modulation mode - center spread

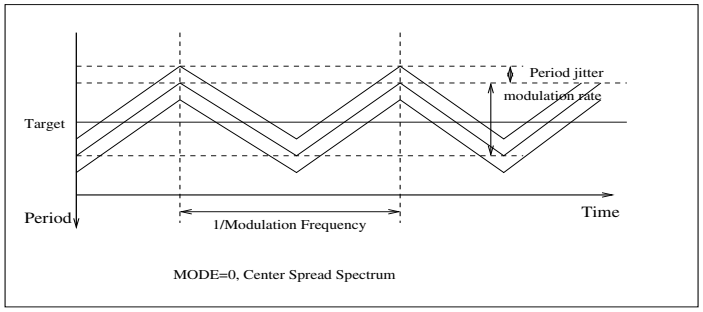
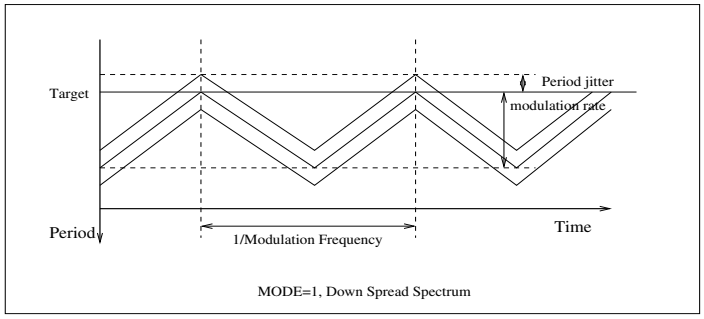


Figure 2-129. Clock modulation mode - down spread



Notes:

1. Modulation frequency is the base frequency at which modulation is done. Supported modulation frequencies are Fin/1024, Fin/2048, and Fin/4096, where Fin in the input clock to PLL macro.
2. Modulation rate is the magnitude of modulation. Supported modulation rates when modulation is enabled are 0.5%, 1%, 2%, 3%, 4%, and 5%. Note that the modulation rate and the divider N settings are inter dependent.

The following restrictions apply.

Table 2-113. Maximum modulation rate versus divider-N settings

N- Divider	Maximum modulation rate
2-63	0% (SSEN = ‘0’)
8-60	0.5%
8-60	1%
8-48	2%
8-31	3%
8-23	4%
8-18	5%

3. Cycle to cycle jitter represents peak to peak cycle jitter on the top of clock modulation, which cannot be avoided due to device noise.

Steps in programming the SSCG PLL interface

This section gives the general steps a programmer must follow while using the SSCG-PLL. Following steps are required to enable PLL:

1. Update profile register for SSCG PLL (SYSC_RUNCKSRER:SSCGPLLEN, SYSC_RUNSSCGCNTR0: SSCGDIVM, SYSC_RUNSSCGCNTR0:SSCGDIVN, SYSC_RUNSSCGCNTR0:SSCGDIVP, SYSC_RUNSSCGCNTR1:SSCGSSEN, SYSC_RUNSSCGCNTR1:SSCGRATE, SYSC_RUNSSCGCNTR1:SSCGMODE, SYSC_RUNSSCGCNTR1:SSCG-FREQ, or corresponding PSS profile register). Update profile register to select PLL clock (SYSC_RUNCKSELR or SYSC_PSSCKSELR). Update the profile register to select the PLL stabilization time (SYSC_SPCCFGR: SSCGSTABS).
2. Set Profile Trigger Register or execute WFI instruction with SYSC_PSSENR:PSSEN = 0xBA.
3. Device state control logic will first enable the SSCG PLL with configured settings, wait for the SSCG PLL stabilization time and then switch the clock source domain to SSCG PLL source.

To change the clock to some other clock:

1. Update the profile register to switch the clock source from SSCG PLL to other clock source. Update the profile register to switched OFF SSCG PLL (SYSC_RUNCKSRER:SSCGPLLEN = ’0’).
2. Set profile trigger register or execute WFI instruction with SYSC_PSSENR:PSSEN = 0xBA.
3. Device state control logic will make sure that SSCG PLL switches OFF only once clock switching is complete.

Steps in programming the GFX PLL interface

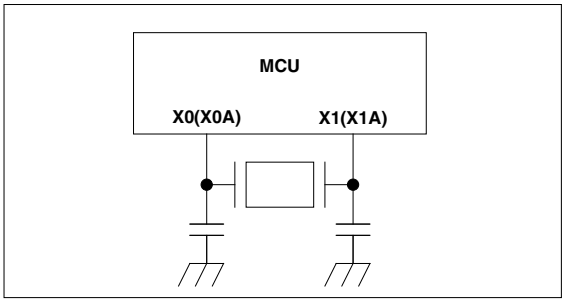
Similar to SSCG-PLL.

2.3.5.6 Oscillator connection

Example of connecting a crystal or ceramic oscillator to the microcontroller

Connect a crystal or ceramic oscillator as shown in Figure 2-130.

Figure 2-130. Connecting a crystal or ceramic Oscillator to the microcontroller



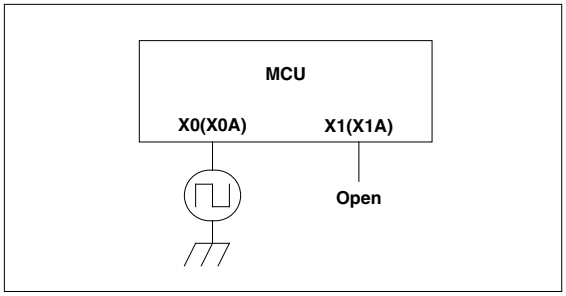
If this connection is used on devices, where the fast clock Input feature is available, the SYSC_SPCCFGR:FCIMEN bit must be set to ‘0’. For more details about the availability of this feature, refer to the datasheet.

Example of connecting an external clock to the microcontroller

As shown in the example in Figure 2-131, connect an external clock to pin X0 (X0A). Pin X1 (X1A) must be open. It is possible to use a high speed clock frequency for the X0 (X0A) pin (consult the datasheet for the actual limits of the clock input frequency). This feature is activated by setting the SYSC_SPCCFGR:FCIMEN bit to ’1’ before switching the device to the external Main clock.

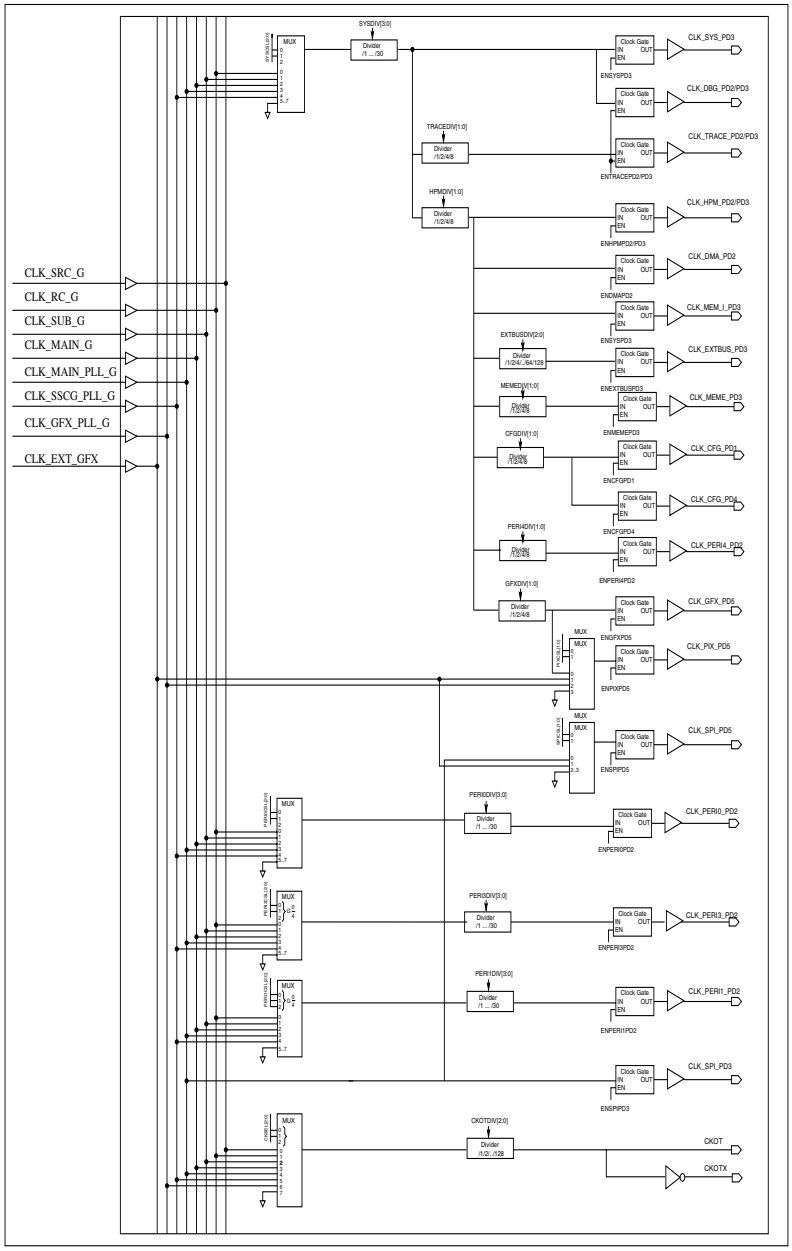
Note: It is recommended to use fast clock Input feature when connecting an external clock to the Main oscillator.

Figure 2-131. Connecting an external clock to the microcontroller



2.3.5.7 Clock selection and distribution block diagram

Figure 2-132. Clock selection and distribution block diagram



2.3.5.8 Clock selection

Clock source after reset

The clock source for all clocks is set to RC clock (CLK_RC) by any hard reset, so that device startup always runs on pre-defined reliable clock to avoid any clock manipulation. This is required from device security point of view.

The RC clock stabilization time of 1920 RC clock cycles is applied after each hard reset.

Clock source after wakeup

It is recommended to set RC clock as system clock in RUN profile before going to PSS, so that device always wakes up with RC clock. If wakeup needs security evaluation (power domain 2 was OFF in PSS), it is mandatory to choose RC clock in RUN profile. This condition is checked by device state control logic. If RC clock is not selected as the system clock during wakeup, profile error NMI is generated. For more details refer to 2.3.2.5 RUN to PSS switching.

Changing the clock source

The clock source selection can be done independently for each clock mux. A clock source switching is done by writing a new value in the SYSC_RUNCKSELR register and applying RUN profile by writing 0xAB to SYSC_TRGRUNCNTR register. A transition to a different clock source is done only when the selected clock is ready (means the corresponding ready flag in SYSC_CKSRESTSR is set).

Similarly, SYSC_PSSCKSELR can be changed any time (while the system is in RUN state), which will be effective once system goes into PSS after applying the new PSS profile settings by writing 0xBA to PSS Switching Enable Register (SYSC_PSSENR:PSSEN) and executing WFI instruction.

Changing clock dividers

The clock divider setting can be done independently for each clock dividers by writing new value in the SYSC_RUNCKDIVR0~2 register and applying RUN profile by writing 0xAB to SYSC_TRGRUNCNTR register.

Similarly, SYSC_PSSCKDIVR0~2 can be changed any time (while the system is in RUN state), which will be effective once system goes into PSS after applying the new PSS profile settings by writing 0xBA to PSS Switching Enable Register (SYSC_PSSENR:PSSEN) and executing WFI instruction.

2.3.5.9 Clock distribution

Table 2-114 gives an overview of all clocks available in the devices and which groups/system components/modules - if available in the device - they supply.

Legend for “Max frequency” column:

At = CY9DF126 (Atlas); A-L = CY9DF125 (Atlas-L); Ti = CY9EF226 (TITAN)

Table 2-114. Clock distribution overview

Clock domain	Clock	Max. frequency (MHz)				Basic components
		Cal	At	A-L	Ti	
CD1	CLK_SYS_PD3	160	128	128	128	Arm Cortex-R4(F) CPU, TCM-SRAM, TCFLASH Interface Note: Flash Clock should not be cut while Flash programming is in progress through CPU.
	CLK_DBG_PD2	160	128	128	128	DAP, Secure Bridge, Secure Checker, APB-Mux, Debug and Trace components(HTM, CTI HTM, CTM, TPIU, Trace Funnel)
	CLK_DBG_PD3	160	128	128	128	ETMR4, CTI ETM
	CLK_TRACE_PD2	160	128	128	128	Debug and Trace component (DAP, ETM, HTM, CTM, CTI HTM, TPIU, Trace Funnel)
	CLK_TRACE_PD3	160	128	128	128	ETMR4
	CLK_HPM_PD2	160	128	128	128	High Performance Matrix (AXI, HPM), AXI RAM
	CLK_HPM_PD3	160	128	128	128	TCFLASH Interface AXI clock
	CLK_CFG_PD4	80	64	64	64	Retention RAM
	CLK_DMA_PD2	160	128	128	128	DMA
	CLK_MEM_I_PD3	160	128	128	128	Interrupt Control, Timing Protection Unit, EEFLASH, BootROM
	CLK_EXTBUS_PD3	160	64	128	128	External Bus Interface
	CLK_MEM_E_PD3	160	128	128	128	HS-SPI
	CLK_CFG_PD1	80	64	64	64	System Controller, Watchdog, RTC, External Interrupt, EICU
	CLK_PERI4_PD2	160	128	128	128	Arbiter, MPUs, USB, Media-LB, Ethernet
	CLK_GFX_PD5	160	-	-	-	Bus clock for Graphics Subsystem
CD2	CLK_PERI0_PD2	80	64	64	64	PERI_0 modules (preferably modulated)
CD3	CLK_PERI1_PD2	40	32	32	32	PERI_1 modules (preferably non- modulated)
CD4	CLK_PERI3_PD2	80	64	64	64	PERI_3 modules (PPU, GPIO etc.).
CD5	CLK_SPI_PD5	160	-	-	-	SPI clock for Graphics Subsystem
CD6	CLK_PIX_PD5	400	-	-	-	Pixel clock for Graphics Subsystem
CD7	CLK_SPI_PD3	160	128	128	128	PLL clock for HS_SPI
CD8	CKOT, CKOTX	160	128	128	128	Clock output function

2.3.6 Source Clock Timers

This device has four independent Source Clock Timers one for each clock source (Slow RC oscillator, RC oscillator, Main clock oscillator, and Sub clock oscillator), which are used as normal timers to issue interrupts at specified intervals and also used to measure the oscillation stabilization time.

Slow RC clock timer

Slow RC clock timer is clocked by the output signal of the internal Slow RC oscillator (CLK_SRC) and can be used as timer when the Slow RC oscillator is enabled. It is also used to measure the Slow RC clock stabilization wait time.

RC clock timer

RC clock timer is clocked by the output signal of the internal RC oscillator (CLK_RC) and can be used as timer when the RC oscillator is enabled. It is also used to measure the RC clock stabilization wait time.

Main clock timer

Main clock timer is clocked by the output signal of the Main oscillator (CLK_MAIN) and can be used as timer when the Main oscillator is enabled. It is also used to measure the Main clock stabilization wait time.

Sub clock timer

Sub clock timer is clocked by the output signal of the Sub oscillator (CLK_SUB) and can be used as timer when the Sub oscillator is enabled. It is also used to measure the Sub clock stabilization wait time.

2.3.6.1 Overview of Source Clock Timer

Source Clock Timer is 16-bit timer used to gate oscillator clock output with configurable time until it is stable for further use. This timer can also be used as a source of periodic interrupt later after source clock is stable.

Features:

- 16-bit up-counter with 16 prescaler configurations
- 4-bit control to the 16 prescaler combination to divide the input clock by $/2^0$ or $/2^1$ or... $/2^{15}$
- Supports two timer operation modes, single-shot mode and continuous mode
- Support for reconfigurable stabilization time by changing compare value during source stabilization time
- Debug support, where timer can be stopped at a break point (debug mode)
- Support for interrupt generation during normal timer operation

2.3.6.2 Block diagram

Figure 2-133. Source Clock Timer block diagram

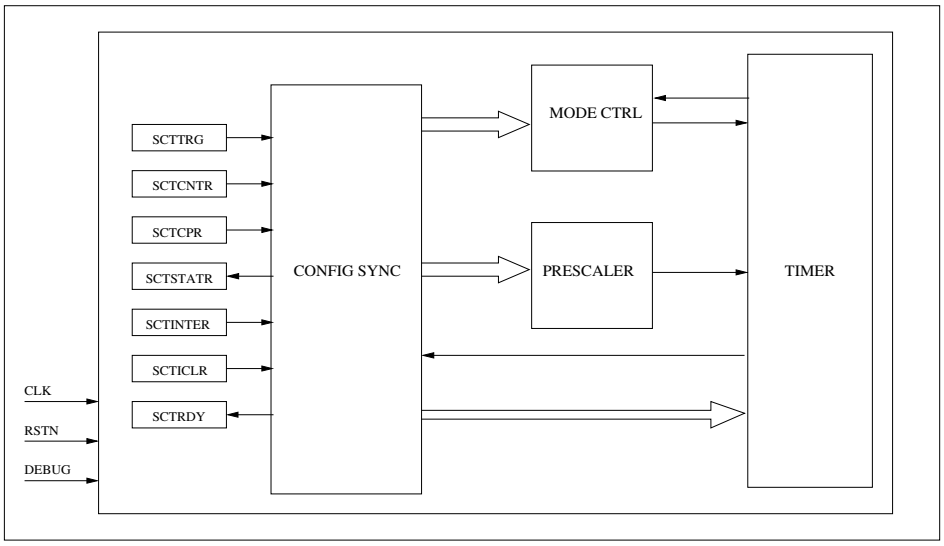


Figure 2-133 shows internal block details of Source Clock Timer. Functionality of each block is described below:

Prescaler

The Source Clock Timer has 4-bit prescale control input to divide the input clock by $/2^0$ or $/2^1$ or... $/2^{15}$. The prescaler is controlled via the configuration register SCTCPR:PSCL. The 16-bit counter runs with the clock provided by the prescaler.

Timer mode control (MODE CTRL)

The Source Clock Timer operates in the following two modes:

1. Single-shot mode: In single-shot mode, when the counter output becomes greater than or equal to the compare value, the counter is loaded with zero and stops incrementing. After oscillator enabling the Source Clock Timer mode is locked to single-shot. Once the clock stabilization time is done the SCTCNTR:MODE can be configured for other timer modes as SCTCNTR register is unlocked for further use.
2. Continuous mode: In the continuous mode, when the counter output becomes greater than or equal to the compare value, the counter is loaded with zero and again starts incrementing.

The mode change is controlled through the configuration register SCTCNTR:MODE.

If SCTCNTR:DBGEN bit is set, SCT will halt in case the MCU is stopped at a break point during debugging. MODE CTRL will hold its value in debug mode and the counter inside Source Clock Timer will be halted.

Clock timer

The clock timer consists of 16-bit counter register, 16-bit comparator. The 16-bit counter is loaded with zero on compare match, else the counter increments for every clock provided by the prescaler block. The 16-bit comparator compares the counter value with SCTCPR:CMPR.

SCT registers are reset to default value at hard reset or when oscillator is disabled. When hard reset is released or oscillator is enabled, the counter is incremented with default values of prescaler and Compare Match Register during clock stabilization, for counting stabilization wait time. During stabilization time when SCTSTATR:BUSY is still high, source clock configuration register SCTCNTR is locked and cannot be programmed by the software. Any write to this register generates PPU error. The default values of prescaler and compare value can be reprogrammed by writing first the correct protected key in System Controller SYSC_PROTKEYR. After this writing new values of SCTCPR:CMPR and SCTCPR:PSCL to extend or shorten the stabilization time by writing into SCTTRG:CGCPT as ‘1’. For description on writing into System Controller registers refer to 2.2.3.1 Protection Key Register (SYSC_PROTKEYR). When SCT counter value is greater than or equal to compare register value, source clock (CLK) is ready to be used as source clock.

After clock stabilization time, SCT can be used as normal timer since SCTCNTR register gets unlocked for normal timer usage.

Interrupt flag can be generated (if SCTINTER:INTE is set) only once during single-shot mode or at regular intervals during continuous mode.

Interrupt flag SCTSTATR:INTF will be set when timer count register value is greater than or equal to comparator value.

SCTSTATR:INTF is cleared by software.

The Source Clock Timer is reset asynchronously when oscillator is disabled.

Configuration synchronization (CONFIG SYNC)

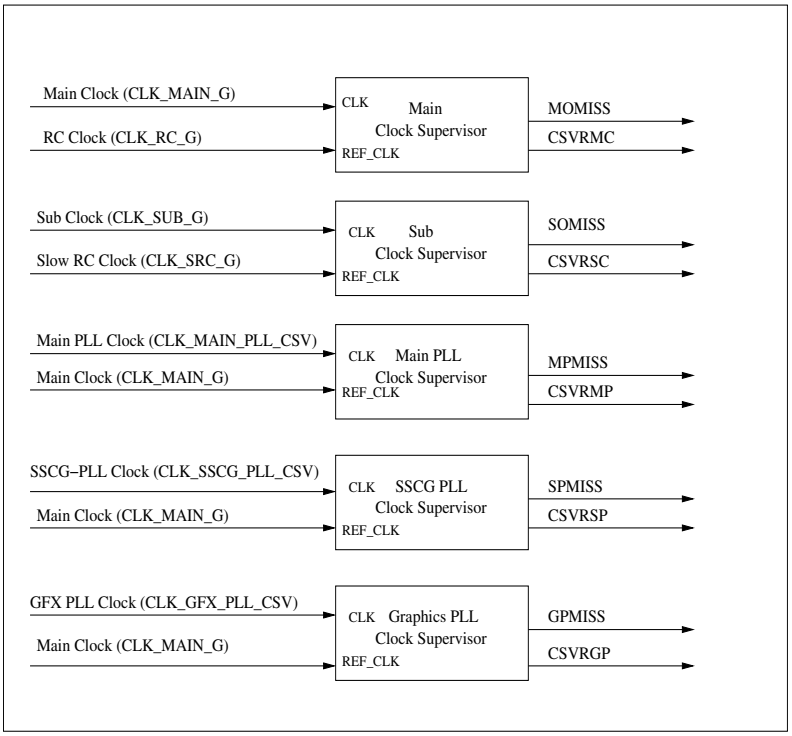
This module synchronizes the configuration data that is updated with the local clock for Source Clock Timer. The values of the configuration register SCTCPR, SCTCNTR, and SCTTRG will be transferred to mode control, prescaler, and clock timer blocks only after synchronization.

Note: Software should not reconfigure the SCT registers too fast, since synchronization of configuration registers to local clock domain takes two clocks of source clocks (which may be very slow compared to bus clock).The synchronization status of the Source Clock Timer can be monitored by reading the SCTSTATR:BUSY bit.

2.3.7 Clock Supervisor

This device has five Clock Supervisor modules, which supervise the frequency range of both external clocks (Main clock, Sub clock) and all PLL clocks (Main PLL, SSCG PLL, GFX PLL). Main clock is supervised with respect to RC clock, so it should be ensured by the user to enable Main oscillator and RC oscillator either at the time of enabling Main Clock Supervisor or before enabling Main Clock Supervisor. Otherwise profile will be discarded and flagged as invalid. Sub clock is supervised with respect to SRC clock, so it is should be ensured by the user to enable Sub oscillator and SRC oscillator either at the time of enabling Sub Clock Supervisor or before enabling Sub Clock Supervisor. All PLLs are supervised with respect to Main clock, so it is should be ensured by the user to enable Main oscillator and corresponding PLL either at the time of enabling PLL supervisor or before enabling PLL supervisor. Otherwise profile will be discarded and flagged as invalid.

Figure 2-134. Clock Supervisor

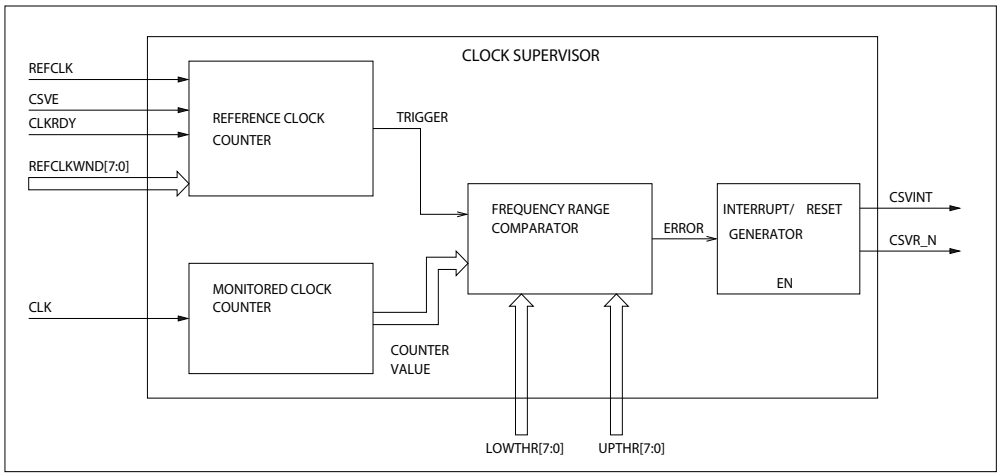


2.3.7.1 Block diagram

Each CSV module responsible for supervising one clock is identical. Clock supervision is done with two counters one running at reference clock (called reference clock counter) and another running at clock to be monitored (called monitored counter). Both counters start at the same time. Once the reference clock counter reaches it configured value (REFCLKWND), it compares the monitored counter value with the lower threshold (LOWTHR) and upper threshold (UPTHR). If the monitored counter value is out of range compared to the upper and lower threshold, it generates a device reset if the clock is used. Otherwise it generates an interrupt depending on the CSV interrupt enable setting in the system setting registers.

The clock and reference clock have to be enabled to enable supervision. Once supervision has been enabled it can only be disabled when the clock has also been disabled.

Figure 2-135. CSV block diagram



Usage of CSV

- At reset all clock supervisors are disabled and inactive during clock stabilization time. User should enable it before using corresponding clock. Once Clock Supervisor is enabled, CSV cannot be disabled while clock is used. Similarly, clock getting monitored cannot be disabled while CSV is ON. While CSV is enabled CSV configurations cannot be changed. This is hardware protected
- CSV can disable its input clock for self testing
- Reset is generated if failing clock is used in device. Interrupt can be generated if failing clock is not used in device
- If interrupt is generated due to failing clock, interrupt will be set again by hardware while clock fails, even if it is cleared by software
- CSV checks the clock usage by clock mux setting in SYSC_CKSELSTSR register. If a failing clock is used only by corresponding SCT, CSV will generate interrupt (not reset)

Main Clock Supervisor

- Main clock is monitored with RC clock
- Main clock frequency range is configurable via SYSC_CSVMOCFGR register
- If clock frequency is out of range and clock is used, then device reset is generated. This can be checked in Reset Cause Registers SYSC_RSTCAUSEUR:CSVRMC and SYSC_RSTCAUSEBT:CSVRMC.
- If clock frequency is out of range and clock is not used then SYSC_SYSERRR:MOMISS bit will be set. Ensure that software does not use this clock later.

Note: RC clock has bigger variation. This has to be considered at programming upper and lower threshold.

Sub Clock Supervisor

- Sub clock is monitored with Slow RC clock
- Sub clock frequency range is configurable via SYSC_CSVSOCFGR register

- If clock frequency is out of range and clock is used, then device reset is generated, which can be checked in Reset Cause Registers SYSC_RSTCAUSEUR:CSVRSC and SYSC_RSTCAUSEBT:CSVRSC
- If clock frequency is out of range and clock is not used then SYSC_SYSERRR:SOMISS bit will be set. Ensure that software does not use this clock later

Note: RC clock has bigger variation. This has to be considered at programming upper and lower threshold.

Main PLL Clock Supervisor

- Main PLL clock is monitored with Main clock
- Main PLL clock frequency range is configurable via SYSC_CSVMPCFGR register
- If clock frequency is out of range and clock is used then device reset is generated, which can be checked in Reset Cause Registers SYSC_RSTCAUSEUR:CSVRMP and SYSC_RSTCAUSEBT:CSVRMP
- If clock frequency is out of range and clock is not used then SYSC_SYSERRR:MPMISS bit will be set. Ensure that software does not use this clock later

SSCG PLL Clock Supervisor

- SSCG PLL clock is monitored with Main clock
- SSCG PLL clock frequency range is configurable via SYSC_CSVSPCFGR register
- If clock frequency is out of range and clock is used then device reset is generated, which can be checked in Reset Cause Registers SYSC_RSTCAUSEUR:CSVRSP and SYSC_RSTCAUSEBT:CSVRSP
- If clock frequency is out of range and clock is not used then SYSC_SYSERRR:SPMISS bit is set. Ensure that software does not use this clock later

GFX PLL Clock Supervisor

- GFX PLL clock is monitored with Main clock
- GFX PLL clock frequency range is configurable via SYSC_CSVGPCFGR register
- If clock frequency is out of range and clock is used then device reset is generated, which can be checked in Reset Cause Registers SYSC_RSTCAUSEUR:CSVRGP and SYSC_RSTCAUSEBT:CSVRGP
- If clock frequency is out of range and clock is not used then SYSC_SYSERRR:GPMISS bit is set. Ensure that software does not use this clock later

Restrictions for the configuration of the UPTHR, LOWTHR and REFCLKWND bits in the Clock Supervisor Configuration Registers

- Requirements for the correct operation of quantization/synchronizers
  - UPTHR > $f_{OBS_max}/2/f_{REF_min}$
  - REFCLKWND > $2(f_{OBS_max}/f_{REF_min})$
  - LOWTHR > $f_{OBS_max}/2/f_{REF_min}$
- Measurement and tolerence requirements to prevent a trigger from occuring
  - $UPTHR/f_{OBS_max} > REFCLKWND/f_{REF_min}$
  - $REFCLKWND/f_{REF_max} > LOWTHR/f_{OBS_min}$

Where:

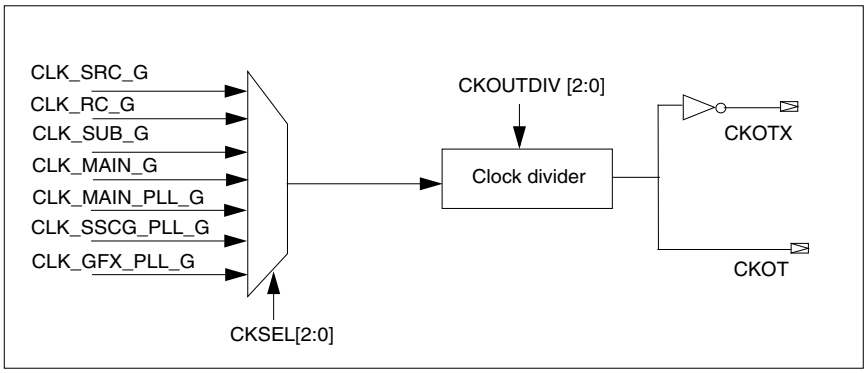
f_{OBS_max} = The maximum frequency of the clock to be observed
f_{OBS_min} = The minimum frequency of the clock to be observed
f_{REF_max} = The maximum reference frequency of the clock
f_{REF_min} = The minimum reference frequency of the clock

Note: Tolerances due to clock jitter are not reflected in the above formulas

2.3.8 Clock output function

The clock output function can be used to output selected internal clocks to CKOT/CKOTX (one non-inverted and one inverted output pin for each clock). The internal clock selection multiplexer is used to select any one of the input clocks.

Figure 2-136. Block diagram of the clock output function



In the Figure 2-136 the SYSC_CKOTCFGR:CKSEL bits are used to select any one of the desired input clock out of all the input clocks. The SYSC_CKOTCFGR:CKOUTDIV bits are used to configure the internal clock divider to divide the selected clock.

For the definition of the clocks that can be selected, see 2.2.15.1 Clock Output Function Configuration Register (SYSC_CKOTCFGR).

2.4 Notes on using System Controller

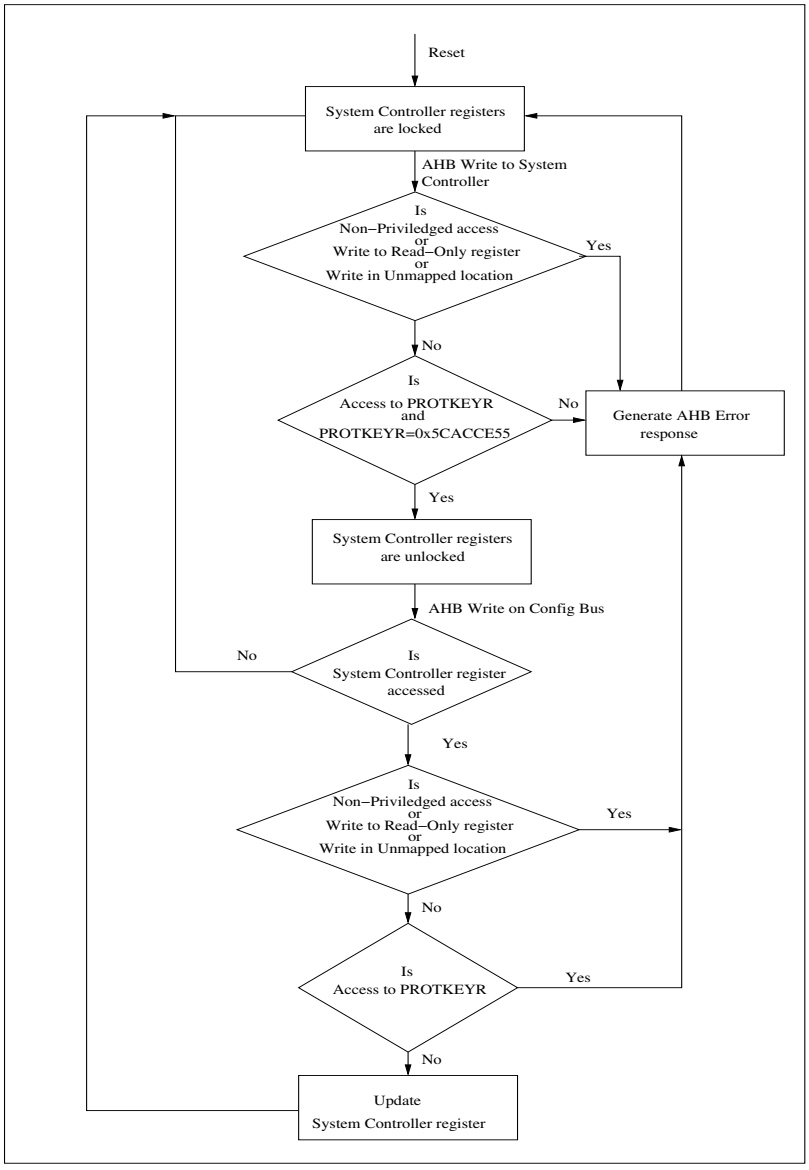
This section is the ‘programmer’s guide’, which lists the usage notes for programming the System Controller module. Programmers are supposed to read these guidelines before programming the System Controller module.

2.4.1 Register configuration

All System Controller register programming is possible only in privileged mode. On top of this protected sequences are used to protect the System Controller registers against overwriting by accident. Registers are protected by a predefined key of size 32-bit, which has to be written to a Compare Key Register in order to unlock the System Controller register set. The write access to the protected register also locks System Controller register set again and clears the Protection Key Register (SYSC_PROTKEYR).

Figure 2-137 shows the System Controller register write sequence.

Figure 2-137. System Controller register write sequence



Note: Source Clock Timer Trigger Register (SCTTRG**)** and Compare Registers (SCTCPR) are protected by protected sequence only when Source Clock Timer is used for clock stabilization check function. When Source Clock Timer is used as a normal timer, the normal PPU functionality is applied for all of Source Clock Timer register.

2.4.2 Device status handling

- Both RUN and PSS profile can be chosen by the user with the constraints as described in 2.3.2.7 Device state profiles. The user should choose the settings based on the performance and power needs of the device
- When the user wants to switch off clock or power domains, user must ensure all the IPs working on that clock or part of that power domain are in idle state and no bus accesses to these IPs are ongoing from any master
- It is recommended to use RC clock as system clock on wake up event i.e. from PSS to RUN state switching. This is to avoid a case where the new requested source clock may miss and cause the device to stay in PSS permanently
- Before applying the profile, check the validity of the profile. This can be checked by the Status Register SYSC_SYSSTSR:IRPARUN, SYSC_SYSSTSR:IRPAPSS and SYSC_SYSSTSR:IPPSPSS bit
- The settings are applied in stages by the device state logic. The user can know current applied state of the profile by polling the Profile Status Register settings. The user can also choose to work in interrupt mode, where device generates status change interrupt, once new profile is applied and device enters into a new state.The interrupt mode is applicable only for RUN state switching
- The profile can be momentarily overridden by Main clock on request from Auto-Calibration module. So the actual state of the device can be known from the Profile Status Register. For Auto Calibration Unit functioning refer to 3. Real Time Clock.
- It is recommended to switch the clock mux to clipped value i.e. tied to zero, if the corresponding clock domain clocks are cut. This ensures that power consumption in the PSS is minimal, when the profiles are being overridden by Auto-Calibration module
- In case of PSS, wakeup event may also trigger IRQ or NMIs based on Interrupt Controller configuration. So the wakeup event may cause CPU switching to reset vector (if CPU is powered down) and later switching to ISR (both maskable and Non-Maskable Interrupts are possible). In case CPU is not powered down (only clock gated) and if wake up event also causes interrupt (i.e. enabled at the Interrupt Controller), CPU comes out of stand by mode and executes ISR. In other case, i.e. wake up event does not cause interrupt (i.e. disabled at the Interrupt Controller), CPU continues in standby mode. For details on exiting conditions from stand by mode of Arm, refer to Arm reference manual

2.4.3 Error handling

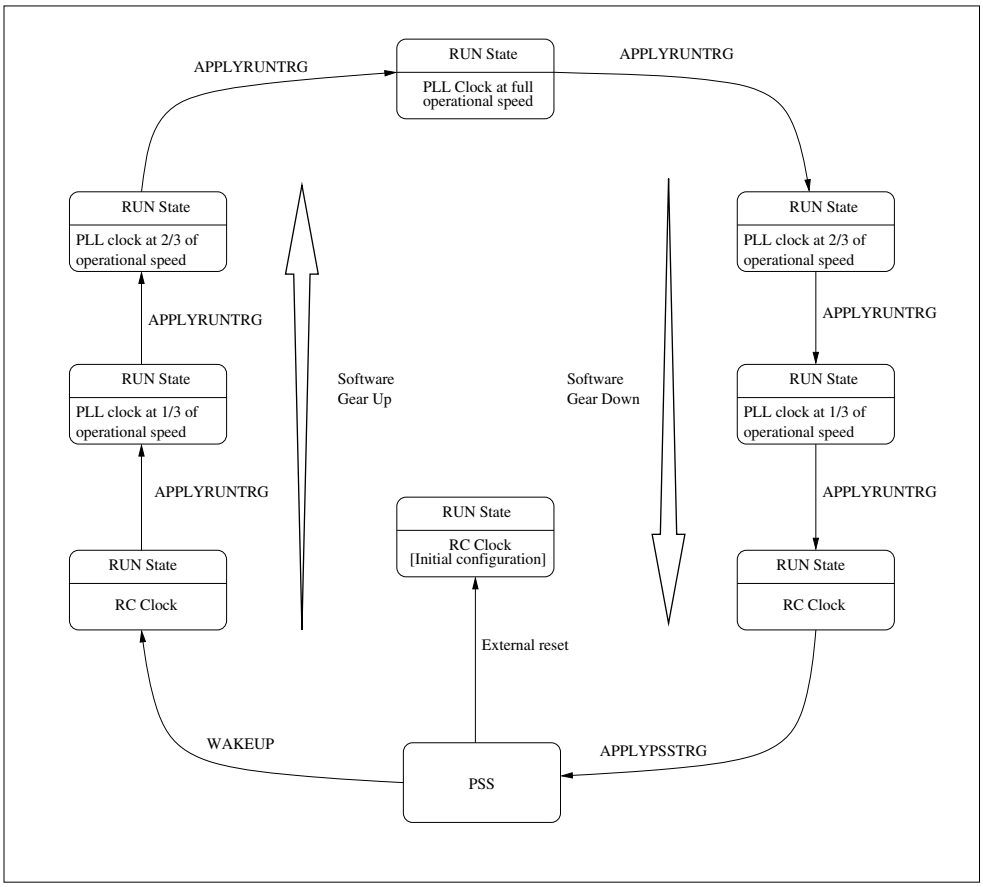
- Device state errors are reported either as reset or interrupts
- All profile related errors are flagged as error interrupt. The profile error includes,
  - Invalid RUN profile for RUN trigger. Error flag SYSC_SYSERRR:RUNERRIF is set
  - Invalid RUN profile for PSS state switching. Error flag SYSC_SYSERRR:RUNWKERRIF is set
  - Invalid PSS profile for PSS state switching. Error flag SYSC_SYSERRR:PSSERRIF is set
  - Invalid RUN trigger value (other than 0xAB) written in SYSC_TRGRUNCNTR register. Error flag SYSC_SYSERRR:RUNTRGEIF is set
  - Invalid PSS Enable value (other than 0xBA) written in SYSC_PSSENR register. Error flag SYSC_SYSERRR:PSSENEIF is set
  - Device state control logic is triggered when profile application is on going. Error flag SYSC_SYSERRR:TRGERRIF
- Source clock frequency range violation can also cause interrupts, if the clocks are not used at that time.The failing clock status is updated in SYSC_SYSERRR register
- Invalidity of RUN profile on wake up event due to bit flipping causes after SYSC_SYSSTSR:RUNBUSY = ’1’ or SYSC_SYSSTSR:PSSBUSY = ’1’ causes profile error reset (PRFERR_N). The cause of this reset is updated in Reset Cause Register. The source clock frequency range violation can also cause reset, if the clocks are used at that time. The cause of this reset is updated in Reset Cause Register
- As device profile is applied in stages by the device state logic, it is possible that device state logic may hang when the enabled source clocks in profile are missing. The system can come out of this hang state by hard resets

2.4.4 Software based clock gearing

The clock gearing down/gearing up may be required to avoid voltage overshoot or undershoots. This also depends on the voltage regulator capability and the state of the device. For the clock gearing steps to be done check the voltage regulator capacity and the device datasheet.

Figure 2-138 shows an example, clock gear handling by the software.

Figure 2-138. Flow chart showing software based clock gearing

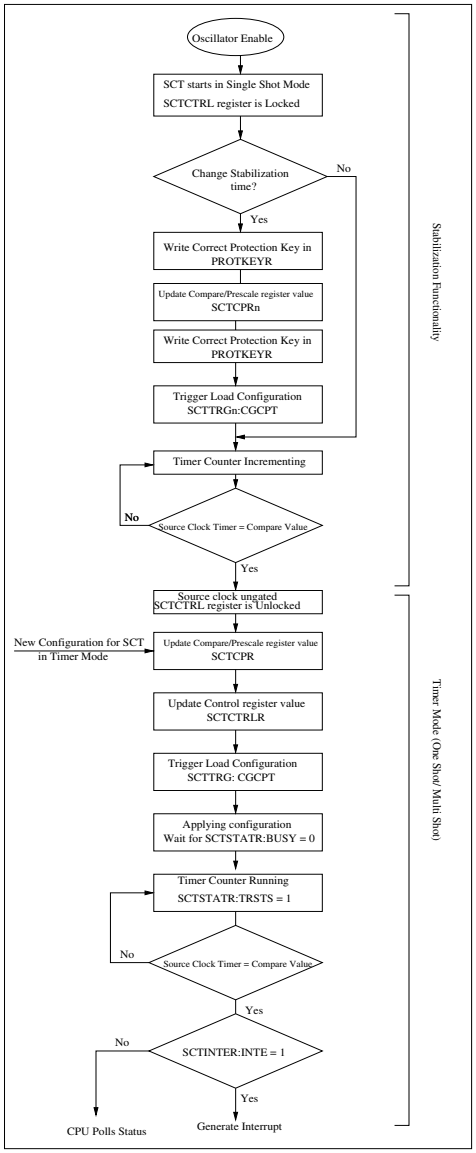


Note: PLL clock gearing needs to be done only by PLL output divider i.e. DIVM setting. This is because only DIVM settings can be changed while PLL is on. Other PLL configurations can be changed only when PLL is disabled.

2.4.5 Notes on using Source Clock Timer

2.3.6 Source Clock Timers gives the general steps a programmer must follow while using the Source Clock Timer module.

Figure 2-139. Source Clock Timer flowchart



2.4.6 Notes on using Clock Supervisor

Notes:

1. While configuring CSV, user should ensure- LOWTHR > (REFCLK_period / CLK_period).
2. Configuration setting condition should be fulfilled UPTHR > LOWTHR.

Main oscillator clock supervision

CSV settings need to be programmed based on Main oscillator frequency, RC oscillator frequency, allowed tolerance on Main oscillator clock, and RC clock inaccuracy.

For example:

Main clock frequency = 4 MHz RC clock frequency = 12 MHz

Max tolerance of Main Osc without reset = 1 MHz to 8 MHz

RC clock inaccuracy = -50% to +100% (6 MHz to 24 MHz)

(Refer datasheet for RC clock frequency and inaccuracy).

Relation between different configuration registers:

Lower threshold = 1/24 * reference window

Upper threshold = 8/6 * reference window

Valid configuration can be:

Reference window = 24

Lower threshold = 1

Upper threshold = 32

Sub oscillator clock supervision

CSV settings need to be programmed based on Sub oscillator frequency, Slow RC oscillator frequency, allowed tolerance on Sub oscillator clock and Slow RC clock inaccuracy.

For example:

Sub clock frequency = 32 kHz

Slow RC clock frequency = 100 kHz

Allowed tolerance of Sub osc = 16 kHz - 64 kHz.

Slow RC clock inaccuracy = -50% to +100% (50 kHz - 200 kHz)

(Refer datasheet for Slow RC clock frequency and inaccuracy).

Relation between different configuration registers:

Lower threshold = 16/200 * Reference window

Upper threshold = 64/50 * Reference window

Valid configuration can be:

Reference window = 25

Lower threshold = 2

Upper threshold = 32

Main PLL clock supervision

CSV settings need to be programmed based on PLL multiplication factor and PLL clock tolerance.

For example:

PLL clock frequency = Main clock freq * divider_N/divider_L

PLL clock tolerance = - 20% to + 20%

Valid configuration can be:

Reference window = divider_L

Lower threshold = 0.8 * divider_N/4

Upper threshold = 1.2 * divider_N/4

Notes:

- CSV setting does not depend on divider_M, since CSV monitors the clock before PLL divider-M
- Division by 4 on lower and upper threshold is there since CSV for PLL runs on prescaled (divide by 4) PLL clock

SSCG/GFX PLL Clock Supervision

CSV settings need to be programmed based on PLL multiplication factor and PLL clock tolerance.

For SSCG PLL:

PLL clock frequency = Main clock freq * SSCG_divider_N * SSCG_divider_P/SSCG_divider_L

PLL clock tolerance = -20% to + 20%

Valid configuration can be:

Reference window = SSCG_divider_L

Lower threshold = 0.8 * SSCG_divider_N * SSCG_divider_P/4

Upper threshold = 1.2 * SSCG_divider_N * SSCG_divider_P/4

Notes:

- CSV setting does not depend on SSCG_divider_M, since CSV monitors the clock before divider- M
- Division by 4 on lower and upper threshold is there since CSV for SSCG PLL runs on prescaled (divide by 4) SSCG PLL clock

3. Real Time Clock

This chapter explains the function and operation of the Real Time Clock.

3.1 Outline of the Real Time Clock

The Real Time Clock (RTC) consists of two modules, the Timer module and the Calibration module. This section lists the features of the Timer and Calibration modules.

The Real Time Clock (RTC) Timer module provides the current real time (HH/MM/SS) and the Calibration module calibrates Sub clock or Slow RC clock with respect to the Main oscillator clock.

Features of the RTC Timer module

The features of the RTC Timer module are:

- It provides counting of real-time, comprising hour, minute and second counters
- It can operate in low-power mode
- Possible source clocks are:
  - Main oscillation (4 MHz)
  - Sub oscillation (32 kHz)
  - RC oscillation (~100 kHz)
- It has a configurable Sub-second counter (RTC time base, normally half-second intervals) to calibrate the RTC for an applied source clock
- Precise source clock switching is synchronised with the Sub-second counter
- Interrupt sources are:
  - Half-second
  - One second
  - One minute
  - One hour
  - One day

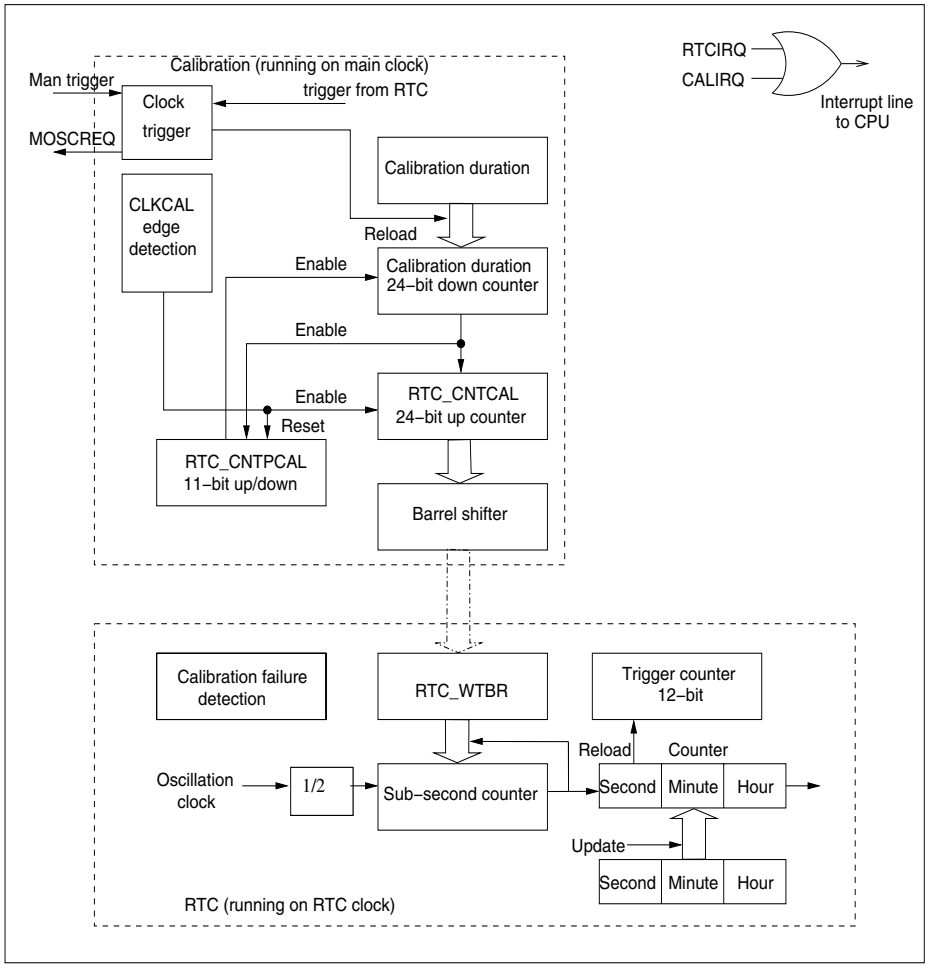
Features of the Calibration module

- The automatic cyclic calibration/measurement of Sub clock (CLK_SUB) or Slow RC clock (CLK_SRC) against Main clock (CLK_MAIN)
- The automatic correction of the RTC Sub-second counter value to keep accurate time
- Low-power modes due to Automatic wake-up/sleep of Main oscillation before/after Calibration phase
- An averaging method to prevent accumulation errors (Automatic modulation of Sub-second counter reload value)
- A configurable calibration interval for cyclic calibration (12-bit counter, counting seconds from RTC)
- A configurable calibration duration (i.e. 250 ms divided by 1/2/4/8/16)
- Automatic/Manual trigger for calibration
- Interrupt sources (to wake up the MCU from low-power mode):
  - Calibration done
  - Calibration failure (missing Main clock)

- Support for MCU debug mode

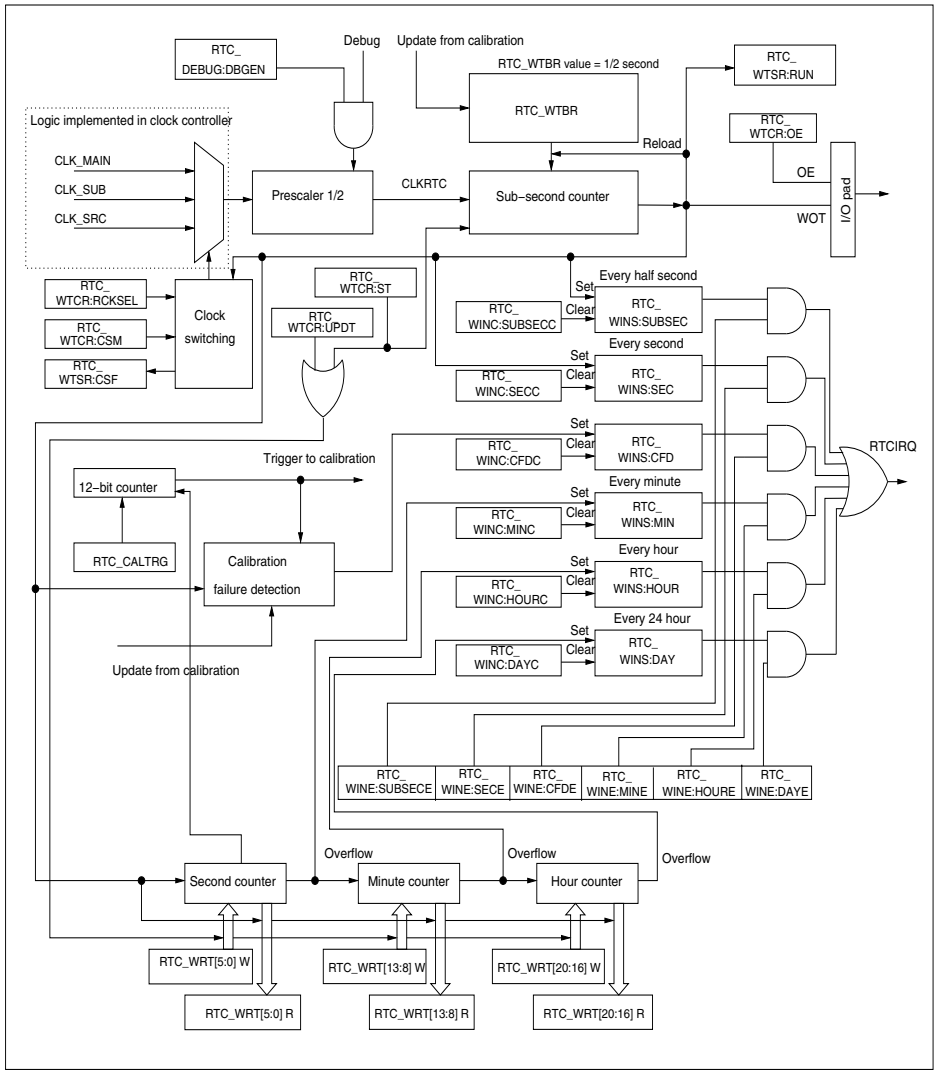
Block diagram of Real Time Clock

Figure 3-1. Block diagram of RTC



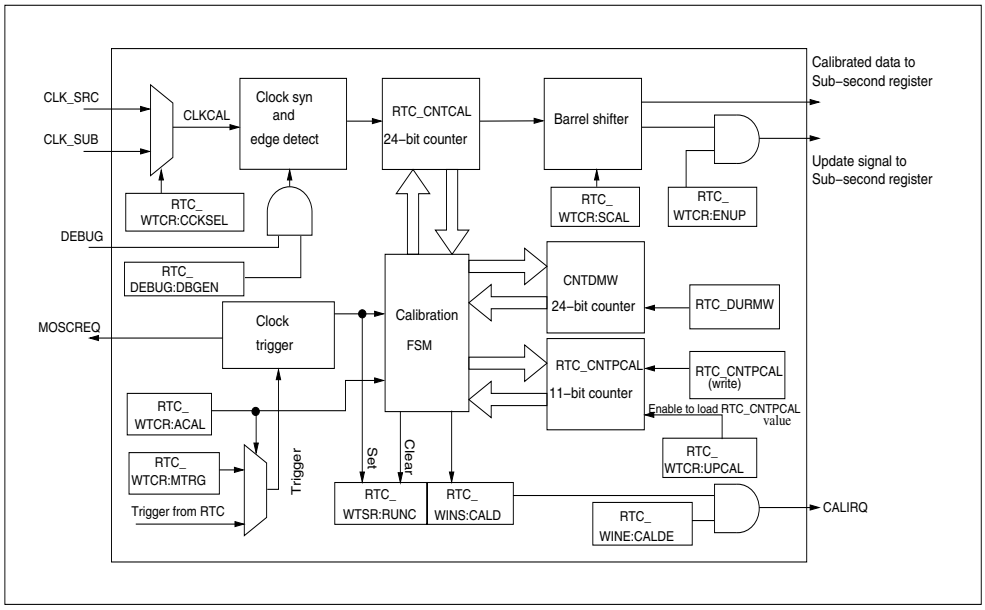
RTC configuration diagram

Figure 3-2. RTC Timer module diagram



Calibration configuration diagram

Figure 3-3. Calibration module diagram



3.2 Real Time Clock registers

This section describes the registers of the RTC in detail.

Registers of the RTC

The following registers are available for the RTC:

- Timer Control Register (RTC_WTCR)
- Timer Status Register (RTC_WTSR)
- Interrupt Status Register (RTC_WINS)
- Interrupt Enable Register (RTC_WINE)
- Interrupt Clear Register (RTC_WINC)
- Sub-Second Register (RTC_WTBR)
- Real Time Register (RTC_WRT)
- Calibration Clock Counter Register (RTC_CNTCAL)
- Calibration Clock Period Counter Register (RTC_CNTPCAL)
- Calibration Duration Register (RTC_DURMW)
- Calibration Trigger Register (RTC_CALTRG)
- Debug Register (RTC_DEBUG)

Memory layout of the RTC registers

Table 3-1. Memory layout of RTC registers

Offset	+3	+2	+1	+0
0x00000000	RTC_WTCR 00000000 00000000 00000000 00000000			
0x00000004	RTC_WTSR 00000000 00000000 00000000 00000000			
0x00000008	RTC_WINS 00000000 00000000 00000000 00000000			
0x0000000C	RTC_WINE 00000000 00000000 00000000 00000000			
0x00000010	RTC_WINC 00000000 00000000 00000000 00000000			
0x00000014	RTC_WTBR 00000000 00000000 00000000 00000000			
0x00000018	RTC_WRT 00000000 00000000 00000000 00000000			
0x0000001C	RTC_CNTCAL 00000000 00000000 00000000 00000000			
0x00000020	RTC_CNTPCAL 00000000 00000000 00000000 00000000			
0x00000024	RTC_DURMW 00000000 00000000 00000000 00000000			
0x00000028	RTC_CALTRG 00000000 00000000 00000000 00000000			
0x0000002C	RTC_DEBUG 00000000 00000000 00000000 00000000			

Note: All RTC registers and counters are initialized only by the power-on reset, low-voltage reset, external reset and Clock Supervisor (CSV) reset.

3.2.1 Timer Control Register (RTC_WTCR)

The Timer Control Register contains various control bits for the RTC including start, stop, mode and clock selection.

Timer Control Register (RTC_WTCR)

Figure 3-4. Timer Control Register (RTC_WTCR)

RTC_WTCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	UPCAL	SCAL[2]	SCAL[1]	SCAL[0]	CCKSEL	ENUP	MTRG	ACAL	read0	read0	RCKSEL[1]	RCKSEL[0]	CSM	UPDT	OE	ST
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp1	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0Wp1	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp1	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-2. Timer Control Register (RTC_WTCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15]	UPCAL	Update Calibration Period Counter This bit is used to update the Calibration Period Counter (RTC_CNTPCAL). By setting this bit to '1', the RTC_CNTPCAL (write) register value is written to the RTC_CNTPCAL counter at the next Calibration trigger (which can be either Automatic or Manual). '0': The update has been completed '1': Update the RTC_CNTPCAL counter with the value of the RTC_CNTPCAL (write) register UPCAL is set by software and cleared by hardware after the completion of an update. Writing '0' by software will not effect this bit. To make sure that update is done properly, the software should not change the RTC_CNTP-CAL (write) register value before UPCAL is de-asserted by hardware. Setting UPCAL to `1' has higher priority than setting it to `0' (in case both updates come at the same time).

Table 3-2. Timer Control Register (RTC_WTCR) bits

Bit Position	Bit Field Name	Bit Description
[14:12]	SCAL	Scale Calibrated Value These bits are used to scale the Calibration value in (RTC_CNTCAL), which is determined by the Calibration module, before it is transferred to the Sub-Second Register. '000': No change '001': Multiply by 2 '010': Multiply by 4 '011': Multiply by 8 '100': Multiply by 16 Others: Prohibited SCAL should be configured w.r.t the value set in the corresponding Calibration Duration Register (RTC_DURMW). For RTC_DURMW = (0.25s)/x, where x = {1,2,4,8,16} Configure SCAL accordingly to match `x' e.g. RTC_DURMW = (0.25s)/2 = 0.125s set SCAL = '001'
[11]	CCKSEL	Clock Select for Calibration This bit is used to select the Calibration clock (CLKCAL). '0': Sub clock (CLK_SUB) is used as CLKCAL '1': Slow RC clock (CLK_SRC) is used as CLKCAL
[10]	ENUP	Enable/Disable Calibration Value Update This bit is used to enable/disable the transfer of the calibration value in Calibration Clock Counter Register (RTC_CNTCAL) to the RTC Sub-Second Register (RTC_WTBR) after the completion of a calibration cycle. '0': Calibration value (RTC_CNTCAL) update to RTC_WTBR is disabled '1': Calibration value (RTC_CNTCAL) update to RTC_WTBR is enabled If enabled, the update of the calibration value to the Sub-Second Register (RTC_WTBR) is carried out in Automatic as well as Manual Calibration mode. The Calibration value in RTC_CNTCAL is scaled according to the configuration of RTC_WTCR:SCAL before updating RTC_WTBR.
[9]	MTRG	Manual Trigger for Calibration This bit is used to trigger a software calibration cycle. '0': No trigger '1': Manual trigger This bit is valid only when the Automatic Calibration mode is disabled (RTC_WTCR:ACAL is `0'). Multiple triggers during calibration are ignored. Reading this bit always returns '0'. This bit is cleared by the hardware at the start of calibration.

Table 3-2. Timer Control Register (RTC_WTCR) bits

Bit Position	Bit Field Name	Bit Description
[8]	ACAL	Automatic Calibration This bit is used to enable/disable Automatic Calibration mode. '0': Automatic Calibration disabled. A manual trigger (RTC_WTCR:MTRG) can be used to trigger calibration '1': Automatic Calibration enabled. The RTC will cyclically trigger the Automatic Calibration. The intervals are configured in the Calibration Trigger Register (RTC_CALTRG) Switching ACAL will reset the ongoing Calibration. In addition changing ACAL from '0' to '1' reloads the Calibration trigger counter with the RTC_CALTRG value **Note:** For a detailed description of the Automatic RTC Calibration, refer to 3.3 Operation of the Real Time Clock and 3.4 Cautions.
[7:6]	read0	-
[5:4]	RCKSEL	Clock Select for RTC These bits are used to select the input clock for the RTC (CLKRTC). '00': Main clock (CLK_MAIN/2) is selected as CLKRTC '01': Sub clock (CLK_SUB/2) is selected as CLKRTC '10': Slow RC clock (CLK_SRC/2) is selected as CLKRTC '11': Prohibited
[3]	CSM	Clock Switching Mode This bit is used to configure the clock switching mode when RTC_WTCR:RCKSEL is changed. '0': Precise switch '1': Immediate switch Precise switch - Clock switching is done when the Sub-second counter reloads from RTC_WTBR i.e. after half second. Immediate switch - Clock switching is done immediately when RTC_WTCR:RCKSEL is changed. Precise clock switching allows the setting of the new RTC_WTBR value w.r.t. the next CLKRTC. To guarantee correct functionality, the CPU must ensure that the new RTC_WTBR and new RCKSEL are set before the Sub-second counter reaches '0' (ideally use the half-second interrupt).
[2]	UPDT	Update This bit is used to update the time value. By setting it to '1', the Second/Minute/Hour register values are written to the Second/Minute/Hour counters respectively. '0': The update has been completed '1': Update the Hour/Minute/Second counters with the values of the Hour/Minute/Second registers, respectively UPDT is set by software and is cleared by hardware after the completion of an update. Writing '0' by software will not effect this bit. Updating by the CPU has a higher priority than updating by hardware (in case both updates come at the same time). Refer to 3.4 Cautions.

Table 3-2. Timer Control Register (RTC_WTCR) bits

Bit Position	Bit Field Name	Bit Description
[1]	OE	Output Enable This bit is used to enable/disable the WOT external pin. '0': The WOT external pin can be used as a General Purpose I/O or for another peripheral block '1': The WOT external pin serves as the output for the Sub-second counter
[0]	ST	Start This bit is used to start/stop the RTC. '0': The Real Time Clock module stops operating, and the Sub-sec- ond counter and the Hour/Minute/Second counters are cleared '1': The settings of the Hour/Minute/Second registers are loaded to the Hour/Minute/Second counters, and the RTC module starts to operate

3.2.2 Timer Status Register (RTC_WTSR)

The Timer Status Register contains various status bits for the RTC.

Timer Status Register (RTC_WTSR)

Figure 3-5. Timer Status Register (RTC_WTSR)

RTC_WTSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RUNC	read0	read0	read0	read0	read0	read0	CSF	RUN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-3. Timer Status Register (RTC_WTSR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	RUNC	Run Calibration This bit indicates the status of the Calibration. '0': The Calibration is inactive '1': The Calibration is in progress This register bit is updated on CLK_MAIN. RUNC is set to '1' when Calibration is started by an Automatic or Manual trigger. RUNC is set to '0' when the Calibration Duration Counter expires or Calibration is reset.
[7:2]	read0	-
[1]	CSF	Clock Switched Flag This bit indicates the status of switching the RTC input clock. '0': The input clock for RTC has not been switched '1': The input clock for RTC has been switched This bit is set by hardware and cleared by software. CSF indicates the switching of the RTC input clock in precise mode (RTC_WTCR:CSM is '0'). When RTC_WTCR:RCKSEL is changed (i.e. the clock on which the RTC will run) in precise mode (RTC_WTCR:CSM is '0'), the clock source will be switched after the Sub-second counter expires. Even though reading RTC_WTCR:RCKSEL will give the updated configuration, the CSF flag will only be set at the actual internal clock switching. To use this bit, it has to be first cleared. Then write the new clock setting to RTC_WTCR:RCKSEL and check if CSF is set to '1'. Update by CPU has higher priority than update by hardware (in case both updates come at the same time).

Table 3-3. Timer Status Register (RTC_WTSR) bits

Bit Position	Bit Field Name	Bit Description
[0]	RUN	RUN This bit indicates the status of the RTC Timer module. '0': The RTC Timer module is inactive '1': The RTC Timer module is active This register bit is updated on CLKRTC. RUN will go low at the second rising edge of the RTC clock after RTC_WTCR:ST has been set to '0'. It will rise again at the second rising edge of RTC clock after RTC_WTCR:ST has been set to '1'.

3.2.3 Interrupt Status Register (RTC_WINS)

The Interrupt Status Register contains various status bits that can trigger an interrupt.

Interrupt Status Register (RTC_WINS)

Figure 3-6. Interrupt Status Register (RTC_WINS)

RTC_WINS																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CALD	CFD	DAY	HOUR	MIN	SEC	SUBSEC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-4. Interrupt Status Register (RTC_WINS) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	CALD	Calibration Done This bit is set when a Calibration cycle has finished. If the CALD interrupt is enabled (RTC_WINE:CALDE is `1'), an interrupt request is generated. '0': Calibration is not triggered or in progress '1': Calibration is done The Calibration Done flag will be set independently of the Calibration value update bit (RTC_WTCR:ENUP). This bit is set to '1' at the Calibration Done event and is cleared by writing '1' to RTC_WINC:CALDC.
[5]	CFD	Calibration Faliure Detection This bit is set when a calibration failure is detected. If the CFD interrupt is enabled (RTC_WINE:CFDE is `1'), an interrupt request is generated. '0': No calibration failure '1': Calibration failure The Calibration Failure Detection flag is set if the RTC_WTBR register is not updated by hardware within one second after an Automatic Calibration trigger event while the Calibration value update feature is enabled (RTC_WTCR:ENUP is `1'). The Calibration Failure Detection feature could also be used in Manual Calibration mode. This requires the issuing of the Manual trigger in time to ensure that the Calibration Duration Counter has elapsed before the next update of the Second counter. The Calibration Failure Detection flag indicates that the Main clock is not available and calibration cannot be executed. This bit is cleared by writing '1' to RTC_WINC:CFDC.

Table 3-4. Interrupt Status Register (RTC_WINS) bits

Bit Position	Bit Field Name	Bit Description
[4]	DAY	Day Flag This bit is set when one day has elapsed. If the DAY interrupt is enabled (RTC_WINE:DAYE is `1'), an interrupt request is generated. '0': No interrupt requests '1': The Hour counter has exceeded 24 hours This bit is set to '1' at one-day intervals, i.e. when the Hour counter overflows. This bit is cleared by writing '1' to RTC_WINC:DAYC.
[3]	HOUR	Hour Flag This bit is set when one hour has elapsed. If the HOUR interrupt is enabled (RTC_WINE:HOURE is `1'), an interrupt request is generated. '0': No interrupt requests '1': The Hour counter has been updated This bit is set to '1' at one-hour intervals, i.e. when the Minute counter overflows. This bit is cleared by writing '1' to RTC_WINC:HOURC.
[2]	MIN	Minute Flag This bit is set when one minute has elapsed. If the MIN interrupt is enabled (RTC_WINE:MINE is `1'), an interrupt request is generated. '0': No interrupt requests '1': The Minute counter has been updated This bit is set to '1' at one-minute intervals, i.e. when the Second counter overflows. This bit is cleared by writing '1' to RTC_WINC:MINC.
[1]	SEC	Second Flag This bit is set when one second has elapsed. If the SEC interrupt is enabled (i.e. RTC_WINE:SECE is `1'), an interrupt request is generated. '0': No interrupt requests '1': The Second counter has been updated This bit is set to '1' at one-second intervals. This bit is cleared by writing '1' to RTC_WINC:SECC.
[0]	SUBSEC	Sub-Second Flag This bit is set when one half second has elapsed. If the SUBSEC interrupt is enabled (RTC_WINE:SUBSECE is `1'), an interrupt request is generated. This bit indicates an interrupt request flag. '0': No interrupt requests '1': The Sub-second counter has expired This bit is set to '1' at half-second intervals, i.e. when the Sub-second counter expires. This bit is cleared by writing '1' to RTC_WINC:SUBSECC.

3.2.4 Interrupt Enable Register (RTC_WINE)

The Interrupt Enable Register contains various enable/disable bits for the interrupts.

Interrupt Enable Register (RTC_WINE)

Figure 3-7. Interrupt Enable Register (RTC_WINE)

RTC_WINE																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CALDE	CFDE	DAYE	HOURE	MINE	SECE	SUBSECE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-5. Interrupt Enable Register (RTC_WINE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	CALDE	Calibration Done Interrupt Enable This bit is used to enable interrupt requests at Calibration done (RTC_WINS:CALD). '0': Interrupt requests at Calibration Done are disabled '1': Interrupt requests at Calibration Done are enabled
[5]	CFDE	Calibration Failure Detection Interrupt Enable This bit is used to enable interrupt requests at Calibration Failure Detection (RTC_WINS:CFD). '0': Interrupt requests at Calibration Failure Detection are disabled '1': Interrupt requests at Calibration Failure Detection are enabled
[4]	DAYE	Day Interrupt Enable This bit is used to enable interrupt requests at one-day (24 hour) intervals (RTC_WINS:DAY). '0': Interrupt requests at one-day (24 hour) intervals are disabled '1': Interrupt requests at one-day (24 hour) intervals are enabled
[3]	HOURE	Hour Interrupt Enable This bit is used to enable interrupt requests at one-hour intervals (RTC_WINS:HOUR). '0': Interrupt requests at one-hour intervals are disabled '1': Interrupt requests at one-hour intervals are enabled

Table 3-5. Interrupt Enable Register (RTC_WINE) bits

Bit Position	Bit Field Name	Bit Description
[2]	MINE	Minute Interrupt Enable This bit is used to enable interrupt requests at one-minute intervals (RTC_WINS:MIN). '0': Interrupt requests at one-minute intervals are disabled '1': Interrupt requests at one-minute intervals are enabled
[1]	SECE	Second Interrupt Enable This bit is used to enable interrupt requests at one-second intervals (RTC_WINS:SEC). '0': Interrupt requests at one-second intervals are disabled '1': Interrupt requests at one-second intervals are enabled
[0]	SUBSECE	Sub-Second Interrupt Enable This bit is used to enable interrupt requests at half-second intervals (RTC_WINS:SUBSEC). '0': Interrupt requests at half-second intervals are disabled '1': Interrupt requests at half-second intervals are enabled

3.2.5 Interrupt Clear Register (RTC_WINC)

The Interrupt Clear Register contains various clear bits for the Interrupt Status Register (RTC_WINS).

Interrupt Clear Register (RTC_WINC)

Figure 3-8. Interrupt Clear Register (RTC_WINC)

RTC_WINC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CALDC	CFDC	DAYC	HOURC	MINC	SECC	SUBSECC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-6. Interrupt Clear Register (RTC_WINC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	CALDC	CALD Interrupt Clear This bit is used to clear interrupt requests at Calibration Done. '0': No effect '1': Clears RTC_WINS:CALD Reading this bit always returns '0'.
[5]	CFDC	CFD Interrupt Clear This bit is used to clear interrupt requests at Calibration Failure Detection. '0': No effect '1': Clears RTC_WINS:CFD Reading this bit always returns '0'.
[4]	DAYC	Day Interrupt Clear This bit is used to clear the Day Interrupt flag. '0': No effect '1': Clears RTC_WINS:DAY Reading this bit always returns '0'.

Table 3-6. Interrupt Clear Register (RTC_WINC) bits

Bit Position	Bit Field Name	Bit Description
[3]	HOURC	Hour Interrupt Clear This bit is used to clear the Hour Interrupt flag. '0': No effect '1': Clears RTC_WINS:HOUR Reading this bit always returns '0'.
[2]	MINC	Minute Interrupt Clear This bit is used to clear the Minute Interrupt flag. '0': No effect '1': Clears RTC_WINS:MIN Reading this bit always returns '0'.
[1]	SECC	Second Interrupt Clear This bit is used to clear the Second Interrupt flag. '0': No effect '1': Clears RTC_WINS:SEC Reading this bit always returns '0'.
[0]	SUBSECC	Sub-Second Interrupt Clear This bit is used to clear interrupt requests at half-second interval. '0': No effect '1': Clears RTC_WINS:SUBSEC Reading this bit always returns '0'.

3.2.6 Sub-Second Register (RTC_WTBR)

The Sub-Second Register stores the reload value for the Sub-second counter that divides the CLKRTC to generate the RTC's time base. The reload value is usually set so that the Sub-second counter will count exactly half a second.

Sub-Second Register (RTC_WTBR)

Figure 3-9. Sub-Second Register (RTC_WTBR)

RTC_WTBR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	WTBR[23]	WTBR[22]	WTBR[21]	WTBR[20]	WTBR[19]	WTBR[18]	WTBR[17]	WTBR[16]	WTBR[15]	WTBR[14]	WTBR[13]	WTBR[12]	WTBR[11]	WTBR[10]	WTBR[9]	WTBR[8]	WTBR[7]	WTBR[6]	WTBR[5]	WTBR[4]	WTBR[3]	WTBR[2]	WTBR[1]	WTBR[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-7. Sub-Second Register (RTC_WTBR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:0]	WTBR	Sub-Second Register The Sub-Second Register holds the value to be reloaded to the Sub-second counter. When the Sub-second counter value becomes '0' the value of WTBR is reloaded to the Sub-second counter. This register is also updated by the calibration circuit when RTC_WTCR:ENUP is set (even in low-power mode). Updating by the CPU has higher priority than updating by hardware (in case both updates come at the same time).

Note: The required value depends on the RTC clock that is used (CLKRTC) and can be calculated with the formula on the following page.

Table 3-8. Example configuration of RTC_WTBR register for different clock configurations

	RTC_WTBR:WTBR[23:0]
Main oscillator, 4 MHz	0x0F423F
RC oscillator, 100 kHz	0x0061A7
Sub oscillator, 32.768 kHz	0x001FFF

Calculation formula:

$$\mathrm{RTC_WTBR:WTBR[23:0]} = \frac{f_{CLKRTC}}{2} \times 0.5\mathrm{s} - 1$$

where

f_{CLKRTC}: CLKRTC frequency [Hz]

3.2.7 Real Time Register (RTC_WRT)

The Real Time Register (RTC_WRT) stores the time information i.e. Second/Minute/Hour values. Reading this register returns the counter values. This register is write accessible; however, the written data is loaded into the counters when the RTC is started (RTC_WTCR:ST set to '1') and each time the RTC_WTCR:UPDT bit is set to '1'.

Real Time Register (RTC_WRT)

Figure 3-10. Real Time Register (RTC_WRT)

RTC_WRT																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WTHR[4]	WTHR[3]	WTHR[2]	WTHR[1]	WTHR[0]	read0	read0	WTMR[5]	WTMR[4]	WTMR[3]	WTMR[2]	WTMR[1]	WTMR[0]	read0	read0	WTSR[5]	WTSR[4]	WTSR[3]	WTSR[2]	WTSR[1]	WTSR[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-9. Real Time Register (RTC_WRT) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:21]	read0	-
[20:16]	WTHR	Hour Register The Hour bits are used to (initially) set the RTC Hour counter value and to read the actual Hour value. The written value is transferred to the Hour counter by setting `1' to the update bit (RTC_WTCR:UPDT) or start bit (RTC_WTCR:ST). Reading the Hour bits returns the latest Hour counter value, not the written value. The Hour counter value is saved to the hour bits every time the Sub-second counter expires, i.e. at intervals of half a second.
[15:14]	read0	-

Table 3-9. Real Time Register (RTC_WRT) bits

Bit Position	Bit Field Name	Bit Description
[13:8]	WTMR	Minute Register The minute bits are used to (initially) set the RTC Minute counter value and to read the actual Minute value. The written value is transferred to the Minute counter by setting `1' to the update bit (RTC_WTCR:UPDT) or start bit (RTC_WTCR:ST). Reading the Minute bits returns the latest Minute counter value, not the written value. The Minute counter value is saved to the minute bits every time the Sub-second counter expires, i.e. at intervals of half a second.
[7:6]	read0	-
[5:0]	WTSR	Second Register The second bits are used to (initially) set the RTC Second counter value and read the actual Second value. The written value is transferred to the Second counter by setting `1' to the update bit (RTC_WTCR:UPDT) or start bit (RTC_WTCR:ST). Reading the Second bits returns the latest Second counter value, not the written value. The Second counter value is saved to the second bits every time the Sub-second counter expires, i.e. at intervals of half a second.

Note: RTC_WRT is initialized only by the power-on reset, low-voltage reset and external resets, and not by any other reset except for unused bits.

3.2.8 Calibration Clock Counter Register (RTC_CNTCAL)

The Calibration Clock Counter Register gives the value of counted CLKCAL cycles of a calibration cycle. The Calibration module controls the Calibration clock counter which is active during the measurement window duration. The counted value can be used to update the Sub-Second Register automatically or manually depending on configurations. Reading this register simply returns the counter value.

Calibration Clock Counter Register (RTC_CNTCAL)

Figure 3-11. Calibration Clock Counter Register (RTC_CNTCAL)

RTC_CNTCAL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	CNTCAL[23]	CNTCAL[22]	CNTCAL[21]	CNTCAL[20]	CNTCAL[19]	CNTCAL[18]	CNTCAL[17]	CNTCAL[16]	CNTCAL[15]	CNTCAL[14]	CNTCAL[13]	CNTCAL[12]	CNTCAL[11]	CNTCAL[10]	CNTCAL[9]	CNTCAL[8]	CNTCAL[7]	CNTCAL[6]	CNTCAL[5]	CNTCAL[4]	CNTCAL[3]	CNTCAL[2]	CNTCAL[1]	CNTCAL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-10. Calibration Clock Counter Register (RTC_CNTCAL) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-

Table 3-10. Calibration Clock Counter Register (RTC_CNTCAL) bits

Bit Position	Bit Field Name	Bit Description
[23:0]	CNTCAL	Calibration Clock Counter Register This register returns the value of the Calibration clock counter. The Calibration clock counter counts full clock cycles of the clock to be calibrated, i.e. CLKCAL. The counting is controlled by the Calibration Module and is active during the Calibration measurement window duration (RTC_DURMW). This unscaled value can be used to manually update the Sub-Second Register. For the Automatic update of RTC_WTBR (RTC_WTCR:ENUP is set to `1'), the value is scaled according to the configured scaling (RTC_WTCR:SCAL) and written to RTC_WTBR. This register always returns the unscaled value. The CNTCAL Register is updated with the Calibration clock counter value at the end of a calibration cycle i.e. when the measurement window duration has elapsed. The Calibration Done flag (RTC_WINS:CALD) and/or interrupt can be used to determine the end of a calibration cycle. **Notes:** 1. For a manual update of the RTC_WTBR register, the software should scale the CNTCAL value w.r.t the configured measurement window duration (RTC_DURMW) and then subtract '1' before writing it to the Sub-Second Register. (Scaling Factor X CNTCAL value) - 1. 2. For more details on the RTC_WTSR:SCAL value calculation, see the description of RTC_WTCR:SCAL.

Note: For a detailed description of the Automatic RTC Calibration, refer to 3.3 Operation of the Real Time Clock and 3.4 Cautions.

3.2.9 Calibration Clock Period Counter Register (RTC_CNTPCAL)

The Calibration Clock Period Counter Register gives the value of the last Calibration clock period of a Calibration cycle. The Calibration clock period counter measures the period length of CLKCAL by counting cycles of the Main clock. The Calibration module controls the Calibration clock period counter which is active during the measurement window duration. The counter is reset with each positive edge of CLKCAL. The counted value at the end of a calibration cycle is used to delay the start of the next calibration duration measurement window. Reading this register simply returns the counter value. Writing this register allows the setting of the Calibration clock period counter.

Calibration Clock Period Counter Register (RTC_CNTPCAL)

Figure 3-12. Calibration Clock Period Counter Register (RTC_CNTPCAL)

RTC_CNTPCAL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CNTPCAL[10]	CNTPCAL[9]	CNTPCAL[8]	CNTPCAL[7]	CNTPCAL[6]	CNTPCAL[5]	CNTPCAL[4]	CNTPCAL[3]	CNTPCAL[2]	CNTPCAL[1]	CNTPCAL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-11. Calibration Clock Period Counter Register (RTC_CNTPCAL) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-

Table 3-11. Calibration Clock Period Counter Register (RTC_CNTPCAL) bits

Bit Position	Bit Field Name	Bit Description
[10:0]	CNTPCAL	Calibration Clock Period Register In this register the value to be loaded to the Calibration clock period counter before a calibration cycle is set and the value of the Calibration clock period counter after a calibration cycle is read. The Calibration clock period counter counts the number of positive edges of CLK_MAIN between two positive edges of CLKCAL within a calibration measurement window duration. This is controlled by the Calibration Module during a calibration cycle. The counter value at the end of the calibration cycle is the remainder of the last CLKCAL period that was not counted by the Calibration clock counter (returned by RTC_CNTCAL). This value is then used for the next calibration cycle. There are separate read and write registers. CNTPCAL (read) register is updated with the Calibration clock period counter value at the end of a calibration cycle. The Calibration Done flag (RTC_WINS:CALD) and/or interrupt can be used to determine the end of a calibration cycle. By setting '1' to RTC_WTCR:UPCAL, the CNTPCAL (write) register value is written to the CNTPCAL counter at the next calibration trigger (which can be either Automatic or Manual). If RTC_WTCR:ACAL is changed during an ongoing calibration, the calibration will be stopped and the CNTPCAL counter will get an intermediate value. To make sure the value is not used for the next calibration cycle, software must update the CNTPCAL counter (e.g. by writing `0').

Note: For a detailed description of the Automatic RTC Calibration, refer to 3.3 Operation of the Real Time Clock and 3.4 Cautions.

3.2.10 Calibration Duration Register (RTC_DURMW)

The Calibration Duration Register stores the reload value of the Calibration Duration Counter (CNTDMW). This value is loaded into the Calibration Duration Counter at the calibration trigger. The value determines the calibration measurement window duration during which CLKCAL is measured.

Calibration Duration Register (RTC_DURMW)

Figure 3-13. Calibration Duration Register (RTC_DURMW)

RTC_DURMW																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	DURMW[23]	DURMW[22]	DURMW[21]	DURMW[20]	DURMW[19]	DURMW[18]	DURMW[17]	DURMW[16]	DURMW[15]	DURMW[14]	DURMW[13]	DURMW[12]	DURMW[11]	DURMW[10]	DURMW[9]	DURMW[8]	DURMW[7]	DURMW[6]	DURMW[5]	DURMW[4]	DURMW[3]	DURMW[2]	DURMW[1]	DURMW[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-12. Calibration Duration Register (RTC_DURMW) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:0]	DURMW	Calibration Duration Register This register stores the reload value of the Calibration duration counter. The counter is controlled by the Calibration Module and is used to determine the measurement window. During the active measurement window CLKCAL is measured by the Calibration clock counter and Calibration clock period counter.

Notes:

1. RTC_DURMW should be programmed w.r.t. CLK_MAIN to achieve a measurement duration of exactly 0.25 s. This would result in a calibration value for CLKCAL, determined by the Calibration clock counter, that perfectly matches the required setting for the Sub-Second Register (RTC_WTBR) i.e. for half a second.
2. Due to the fixed clock prescaler of 2 for the RTC clock (CLKRTC), measuring for 0.25 s returns a calibration value for 0.5 s.
   Shorter measurement window durations are possible, but should be a half, quarter, etc., of the normal duration of 0.25 s. By configuring the scaling feature accordingly (RTC_WTCR:SCAL), the RTC_CNTCAL value of shorter measurement windows can be scaled to match the Sub- second counter requirements for 0.5 s.

Table 3-13. Example configuration of RTC_DURMW register for different RTC_WTCR:SCAL configurations at f_{CLK_MAIN}, 4 MHz

RTC_WTCR:SCAL	RTC_DURMW: DURMW[23:0]
'000'	0x0F4240
'001'	0x07A120
'010'	0x03D090
'011'	0x01E848
'100'	0x00F424

Calculation Formula:

$$RTC_DURMW:DURMW[23:0] = \frac{f_{CLK_MAIN} \times 0.25s}{2^{SCAL}} - 1$$

where,

f_{CLK_MAIN}: CLK_MAIN frequency [Hz]

3.2.11 Calibration Trigger Register (RTC_CALTRG)

The Calibration Trigger Register stores the reload value of the 12-bit Calibration trigger counter, which is used to generate cyclic trigger events for Automatic Calibration. This register defines the time interval in seconds between two consecutive calibration cycles (In fact, the actual time interval is RTC_CALTRG+1). In low-power mode, the counter cyclically triggers the Calibration Module to switch on the Main oscillation and execute a calibration cycle.

Calibration Trigger Register (RTC_CALTRG)

Figure 3-14. Calibration Trigger Register (RTC_CALTRG)

RTC_CALTRG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CALTRG[11]	CALTRG[10]	CALTRG[9]	CALTRG[8]	CALTRG[7]	CALTRG[6]	CALTRG[5]	CALTRG[4]	CALTRG[3]	CALTRG[2]	CALTRG[1]	CALTRG[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-14. Calibration Trigger Register (RTC_CALTRG) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:12]	read0	-
[11:0]	CALTRG	Calibration Trigger Register This register stores the reload value of the 12-bit programmable Calibration trigger counter. In Automatic Calibration mode (RTC_WTCR:ACAL is set to `1'), the Calibration trigger counter is decremented on each RTC Second interval. When the counter reaches zero, a calibration trigger is issued to the Calibration Module. In Manual Calibration mode, this timer is not used.

Note: While the actual time interval (in seconds) between two consecutive calibration cycles is RTC_CALTRG:CALTRG [11:0] +1, the register value is actually RTC_CALTRG :CALTRG [11:0] -1.

3.2.12 Debug Register (RTC_DEBUG)

The Debug Register has configuration bits for debug functionality.

Debug Register (RTC_DEBUG)

Figure 3-15. Debug Register (RTC_DEBUG)

RTC_DEBUG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-15. Debug Register (RTC_DEBUG) bits

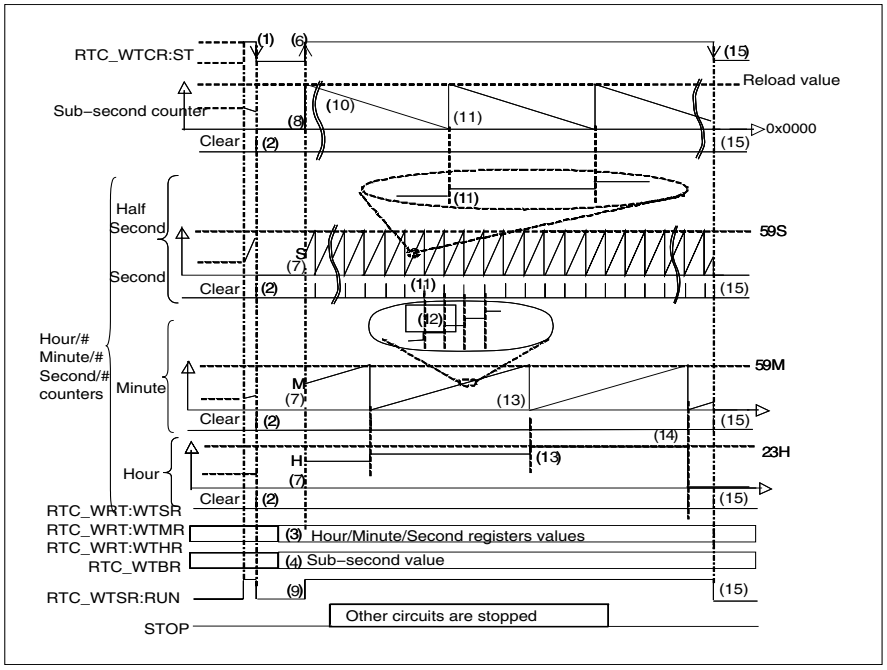
Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	DBGEN	Debug Enable This bit is used to enable/disable Debug mode for the RTC. '0': Debug mode disabled '1': Debug mode enabled When DBGEN is set to '1' and the processor is in the debug state, CLKRTC, CLKCAL, and CLK_MAIN are masked. Therefore the Sub-second counter and Calibration state machine are stopped. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.

3.3 Operation of the Real Time Clock

This section describes the RTC Timer module and Calibration module operation.

RTC Timer module operation

Figure 3-16. RTC Timer module operation



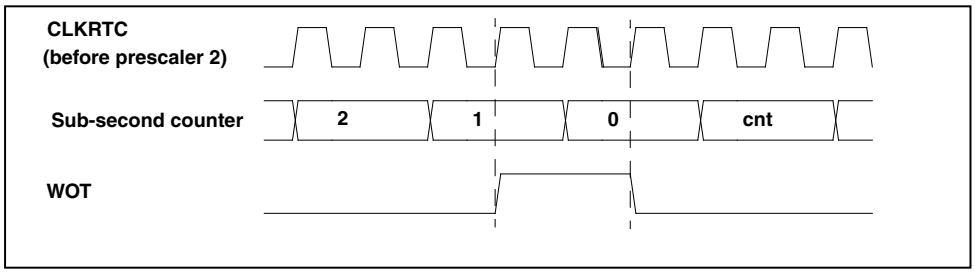
The operation of the RTC Timer module, shown in Figure 3-16, is described as follows:

1. The software writes the start bit RTC_WTCR:ST to '1' and then '0'. (register initialization operation)
2. This resets the Sub-second counter to '0'. While RTC_WTCR:ST is '0' all counters for Sub- second, Second, Minute, and Hour are stopped.
3. The software writes Hour, Minute, and Second values to the Real Time Register, RTC_WRT.
4. The software writes the appropriate values to the Sub-Second Register, RTC_WTBR.
5. The interrupt request bits (RTC_WINS:SEC, RTC_WINS:MIN, RTC_WINS:HOUR, RTC_WINS:DAY, and RTC_WINS:SUBSEC) are cleared to '0', and the interrupt request enable bits (RTC_WINE:SECE, RTC_WINE:MINE, RTC_WINE:HOURE, RTC_WINE:DAYE and RTC_WINE:SUBSECE) are set to '1' i.e. interrupts enabled.
6. The software writes the start bit RTC_WTCR:ST to '1' to start the RTC counting operation.
7. This (RTC_WTCR:ST = '1') causes the values of the Real Time Register RTC_WRT to be loaded to the Hour/Minute/Second timers.
8. The value of the Sub-Second Register RTC_WTBR is loaded to the Sub-second counter.
9. With the second rising edge of CLKRTC the run flag RTC_WTSR:RUN is set to '1'.
10. The Sub-second counter begins counting on the selected CLKRTC divided by 2. This could be the Main clock (CLK_MAIN) divided by 2 (4/2 MHz), Sub clock (CLK_SUB) divided by 2 (32.768/2 kHz) or Slow RC clock (CLK_SRC) divided by 2 (100/2 kHz).
11. When the Sub-second counter reaches '0', the value of the Sub-Second Register is loaded to the Sub-second counter, generating a sub-second interrupt request, i.e. the RTC_WINS:SUBSEC flag is set. For each other half-second interrupt, a one-second interrupt is generated, i.e. the RTC_WINS:SEC flag is set, and the Second counter is incremented.
12. When the Second counter reaches 59, it is cleared the next time it counts up, at which point the Minute counter counts up, generating a 1-minute interrupt request.

13. When the Minute counter reaches 59, it is cleared the next time it counts up, at which point the Hour counter counts up, generating a 1-hour interrupt request.
14. When the Hour counter reaches 23, it is cleared the next time it counts up, at which point a 1- day interrupt request is generated.
15. Writing RTC_WTCR:ST to '0' resets and stops the Sub-second counter and the Hour/Minute/ Second counters.

Note: To operate the RTC in low-power mode, the selected CLKRTC must remain enabled in the System Controller's Power Saving State (PSS) profile.

Figure 3-17. WOT pin operation



Calibration operation

The Calibration module provides automatic (and manual) RTC calibration. The module is able to operate in all run modes of the MCU if configured accordingly. The Main oscillation clock is used as an accurate time reference signal to measure the (normally) less accurate clock used for the RTC (CLKCAL). If the Main oscillator is switched off (e.g. during low-power mode), the Calibration module will temporarily switch it on, execute the calibration and switch it off afterwards to ensure low-current consumption.

Two counters operate during a calibration cycle to measure CLKCAL. The Calibration clock counter measures full clock cycles of CLKCAL and returns the value in RTC_CNTCAL. The second counter measures the period length of CLKCAL by counting Main clock cycles between two rising edges. This value is returned in RTC_CNTPCAL. Both counters operate during a configurable measurement window duration (RTC_DURMW), which is normally set up for 0.25 s. This allows the counted CLKCAL cycles to be directly used as the reload value for the Sub-second counter. During Automatic Calibration mode this value is transferred to the Sub-second counter by hardware means only, i.e. no software interaction is required.

Due to the cyclic and automatic re-calibration of the RTC Sub-second counter, any static and dynamic deviation of the RTC clock can be compensated for, resulting in very accurate RTC timing behavior. In the Calibration module, a novel method that positions the measurement window relative to CLKCAL actually modulates the Sub-second counter. This modulation can compensate for deviations that normally accumulate over time (caused by the limited resolution of the Sub- second counter to count only full CLKRTC cycles).

The following steps describe how to set up the Calibration module. The calibration operation is shown in Figure 3-18:

1. Configure the measurement window duration time for a calibration cycle in RTC_DURMW (normally 1,000,000 for 0.25 s and 4 MHz Main clock). During this time the clock to be calibrated (CLKCAL) is measured. The longer the measurement window, the higher the accuracy of the Calibration value i.e the RTC timing.
2. Set the Calibration Value Scaling (RTC_WTCR:SCAL) with respect to the value set in the RTC_DURMW. For the normal measurement window of 0.25 s, scaling is not required (RTC_WTCR:SCAL must be set to '0'). Select the clock to be calibrated by configuring RTC_WTCR:CCKSEL.
3. Set RTC_WINE:CALDE to '0' or '1' to disable or enable interrupt generation after a calibration cycle has finished. This may wake up the MCU in low-power mode.

Automatic Calibration:

4. Write the Calibration trigger counter value to RTC_CALTRG to configure the time interval between two calibration cycles. For low-power mode, this also defines the intervals to wake up the Main oscillator.
5. Set RTC_WTCR:ENUP to '1' to enable the automatic transfer of the measured calibration value to the Sub-Second Register (RTC_WTBR) at the end of every calibration cycle.

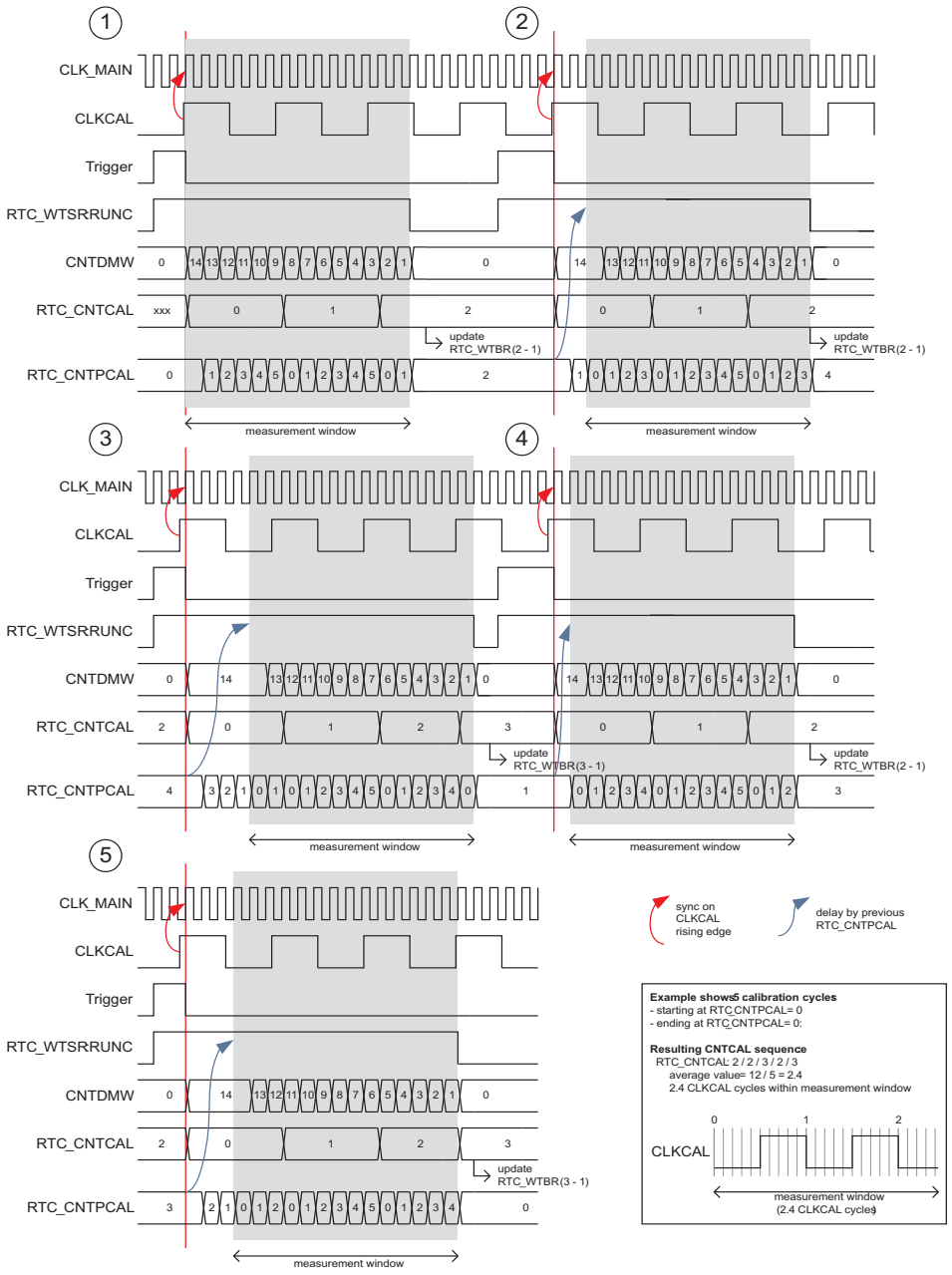
6. Set RTC_WTCR:ACAL to '1', to enable Automatic Calibration. When the RTC is operating (RTC_WTSR:RUN = '1', the Calibration trigger counter will run and generate cyclic calibration trigger events.
7. Set RTC_WINE:CFDE to '1' to enable interrupt generation (and MCU wake-up) in case of a calibration failure. An interrupt request will be generated when a calibration failure is detected if RTC_WTBR is not updated within one second after the hardware trigger. This happens when the Main clock is missing.

Manual Calibration:

4. Set RTC_WTCR:ACAL to '0' to disable Automatic Calibration and allow the calibration to be triggered by a Manual trigger.
5. Set RTC_WTCR:ENUP to '0' or '1' to disable or enable the updating of the Sub-Second Register (RTC_WTBR) with the calibration value at the end of the Manual Calibration.
6. Clear the Calibration done flag (RTC_WINS:CALD) by writing RTC_WINC:CALDC to '1'.
7. Set RTC_WTCR:MTRG to '1' - this triggers the calibration by software.
8. If RTC_WTCR:ENUP is disabled, read the RTC_CNTCAL value after a Calibration-done interrupt. If necessary, scale the read value according to the programmed Calibration duration value. Write the scaled value to the RTC_WTBR register.
9. If RTC_WTCR:ENUP is enabled, the Calibration module will scale the value according to the configuration in RTC_WTCR:SCAL and update.

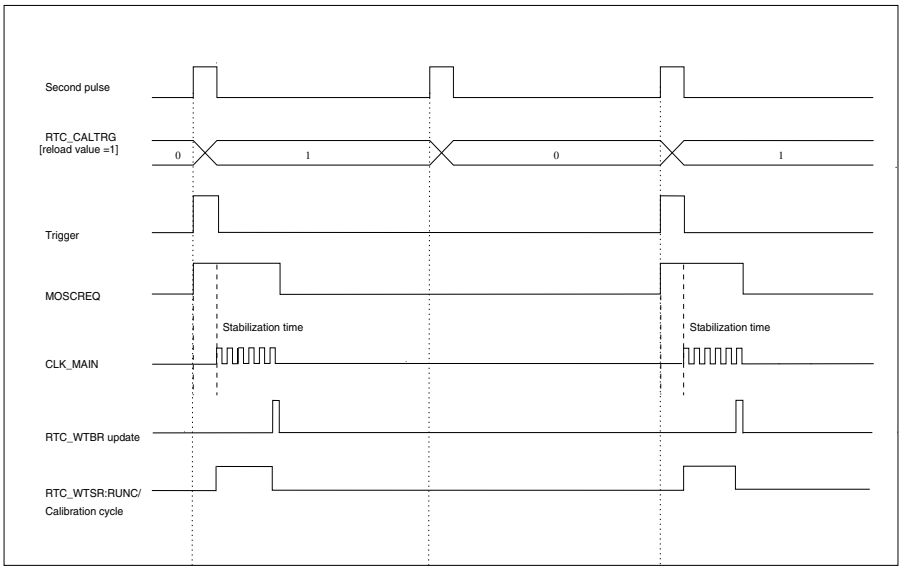
Note: For Automatic Calibration in low-power mode, ensure a valid stabilization time for CLK_MAIN oscillation is configured. Information about stabilization time can be found in 2. System Controller.

Figure 3-18. Calibration operation



Note: Simplified drawing without oscillation stabilization time.

Figure 3-19. Cyclic Calibration



Cyclic Calibration

Cyclic Calibration, as shown in Figure 3-19, is done in Automatic Calibration mode. The Calibration trigger counter reloads the value configured in RTC_CALTRG and decrements on every second pulse from the RTC. It generates a calibration trigger when it expires. Then Main oscillator start request (MOSCREQ) is asserted to switch on the Main oscillation, which is available after a specified stabilization time. The calibration now operates with the Main clock (CLK_MAIN). At the end of a Calibration cycle and with Automatic RTC_WTBR update enabled (RTC_WTCR:ENUP = '1'), RTC_WTBR will be updated with the measured (and scaled) value of RTC_CNTCAL. The Main oscillator start request (MOSCREQ) deasserts at the end of the calibration to switch off the Main oscillator in low-power mode. As the Calibration trigger counter reloads after it expires, the Calibration operation repeats cyclically. If the Main clock is missing after MOSCREQ is asserted, a Calibration Failure Detection (CFD) interrupt is generated (if RTC_WINE:CFDE = '1'). The Calibration Trigger Register (RTC_CALTRIG) value defines the duration between two Calibrations.

3.4 Cautions

This section describes the cautions to be considered when using the RTC.

Cautions

- If '1' is written to the update bit (RTC_WTCR:UPDT) at the same time the update completes, the update bit (RTC_WTCR:UPDT) is set to '0'
- When the Second counter holds a value of 59 even if '1' is written to the update bit (RTC_WTCR:UPDT), the Hour/Minute/Second counters are not updated, and the update bit remains '0'. In order to update the Hour/Minute/Second counters:
  - Write '0' to the start bit (RTC_WTCR:ST)
  - Clear the Hour/Minute/Second registers to ‘0’
  - Write '1' to the start bit (RTC_WTCR:ST)
- If the peripheral clock is stopped after updating the Hour/Minute/Second counters using the update bit (RTC_WTCR:UPDT), read the Hour/Minute/Second Registers to confirm that they have been updated before stopping the peripheral clock
- To reset the Hour, Minute, Second and Sub-second counters before starting the RTC
  - Write '1' to the start bit (RTC_WTCR:ST)
  - Wait until RUN (RTC_WTSR:RUN) is set to '1'
  - Write '0' to the start bit (RTC_WTCR:ST)
- The RTC module Sub-Second Register stores the reload value for the Sub-second counter. This value is reloaded after the Sub-second counter reaches '0'. When modifying the Sub-Second Register, ensure that a reload operation is not performed during the write operation. However, if a Sub-Second Register update is done immediately after an RTC second interrupt, there should be enough time to securely modify the registers until the next reload operation (next second interrupt) even if RTC_WTCR:ST is not set to '0' and the module is in operation
- If a reload has occurred during the updating of the Sub-Second Register (RTC_WTBR), an unexpected value may be reloaded to the Sub-second counter. Therefore, the Sub-Second Register (RTC_WTBR) should be updated with the start bit (RTC_WTCR:ST) set to '0'
- When updating the Hour/Minute/Second registers using the RTC_WTCR:ST bit, the following must be taken into account: The new value is written into the registers with the rising edge of the RUN bit. This RUN bit is clocked by the CLKRTC/2. To ensure that the update is done properly, write the new values into the registers, set RTC_WTCR:ST to '0', wait for the RUN bit to go low and then start the circuit again by setting RTC_WTCR:ST to '1'. RUN will go low at the second rising edge of the CLKRTC after RTC_WTCR:ST has been set to '0'. It will rise again at the second rising edge of CLKRTC after RTC_WTCR:ST has been set to '1'. If this operation is to be repeated several times directly after each other, wait for RUN to go high before setting RTC_WTCR:ST to low again
- If the Sub-Second Register (RTC_WTBR) is set to '0', the Sub-second counter does not operate, causing the RTC module to also not operate
- If a carry operation occurs (e.g. Second counter overflows and Minute counter increments) while reading the Real Timer Register (RTC_WRT), inconsistent values may be read. To avoid this, the interrupts (RTC_WINS:SUBSEC, RTC_WINS:SEC, RTC_WINS:MIN, RTC_WINS:HOUR, RTC_WINS:DAY) should be used to read the time (HH/MM/SS)

4. Watchdog Timer

This chapter explains the function and operation of the Watchdog Timer.

4.1 Outline of the Watchdog Timer

This section describes the features and block diagram of the Watchdog Timer.

Features of the Watchdog Timer

The Watchdog Timer module implements the window Watchdog functionality. It uses a separate window for the RUN and Power Saving States (PSS) defined for the device. A Watchdog reset is generated when the Watchdog is not cleared in the window, and depending on the configuration, a Prewarn interrupt may be generated before the reset. Watchdog functionality is enabled only in device user mode and disabled in board test mode. This module is always in the 'ON' domain.

The features of the Watchdog Timer are:

- A 32-bit Watchdog counter to implement Watchdog functionality
- It supports the selection of a Watchdog clock for operation from four possible choices: Main clock (CLKMAIN); Sub clock (CLKSUB); RC clock (CLKRC); and Slow RC clock (CLKSRC)
- The Watchdog Timer module works in either RUN state or PSS, the choice of which corresponds to the device state. Each state has its own configuration. It supports Window Watchdog functionality
- A unique trigger sequence is implemented to clear the Watchdog counter (refer to ‘Triggering Watchdog’)
- It supports reset or Non-Maskable Interrupt (NMI) generation in the event of Watchdog errors (refer to ‘Watchdog error’)
- A Prewarn interrupt can be generated in the event of a Watchdog error before a Watchdog reset or NMI are generated (refer to ‘Generation of prewarn interrupt and reset/NMI’)
- Additional safety is implemented in the form of three times redundancy and error correction logic for important configuration bits
- The configuration to trigger the Watchdog is possible in privileged or non-privileged mode, depending on the Peripheral Protection Unit (PPU) setting
- It supports a protection sequence for configuration register programming
- It supports the checking of the Watchdog counter status, which is ouput from the Microcontroller Unit (MCU) (refer to ‘Observe output’)
- It supports the stopping of the Watchdog counter if the MCU reaches a break point
- It supports the prevention of register reconfiguration after the Watchdog Configuration Register LOCK bit (WDG_CFG:LOCK) is set (refer to 4.2.15 Watchdog Configuration Register (WDG_CFG))
- The Watchdog counter is reset using a soft reset

Watchdog important key list

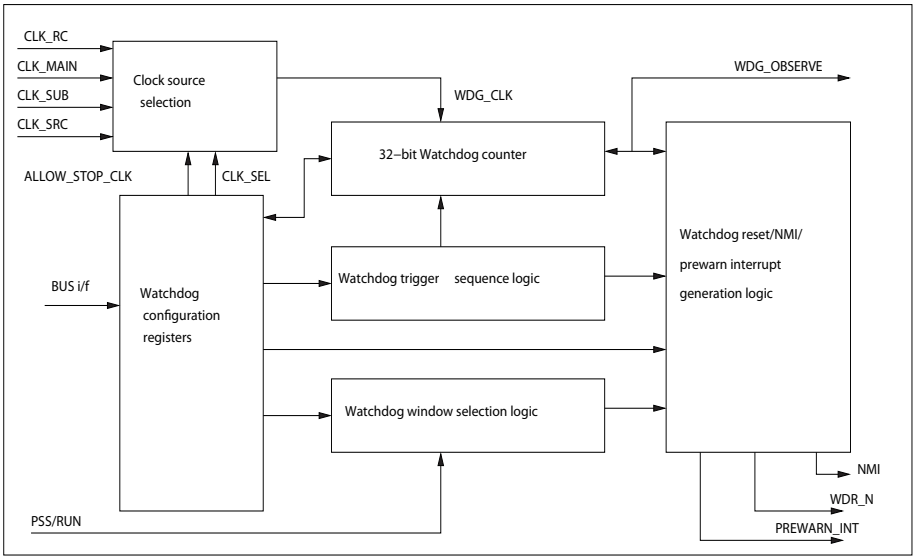
Table 4-1. Watchdog Protection Register unlock key

Key value	Relevant register	Purpose
0xEDACCE55	WDG_PROT:KEY[31:0]	Watchdog register unlock key value

Block diagram of the Watchdog Timer

In Figure 4-1, DEVSTAT indicates the state of the Central Processing Unit (CPU) (e.g. whether the CPU is in RUN state or PSS). The Watchdog reset (WDR_N) is one of the hard reset sources of the System Controller. For details of PREWARN_INT and NMI refer to 4.3.2 Watchdog operation. WDOG_OBSERVE is output by the MCU so that the running state of the Watchdog counter can be externally observed.

Figure 4-1. Block diagram of Watchdog Timer



4.2 Watchdog Timer registers

This section describes the registers of the Watchdog Timer.

Registers of the Watchdog Timer

The following registers are available for the Watchdog Timer:

- Watchdog Protection Register (WDG_PROT)
- Watchdog Counter Register (WDG_CNT)
- Watchdog Reset Cause Register (WDG_RSTCAUSE)
- Watchdog Trigger 0 Register (WDG_TRG0)
- Watchdog Trigger 1 Register (WDG_TRG1)
- Watchdog Interrupt Configuration Register (WDG_INT)
- Watchdog Interrupt Clear Register (WDG_INTCLR)
- Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG)
- Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG)
- Watchdog Lower Limit RUN Register (WDG_RUNLL)
- Watchdog Upper Limit RUN Register (WDG_RUNUL)
- Watchdog Lower Limit PSS Register (WDG_PSSLL)
- Watchdog Upper Limit PSS Register (WDG_PSSUL)
- Watchdog Reset Delay Counter Register (WDG_RSTDLY)
- Watchdog Configuration Register (WDG_CFG)

Memory layout of Watchdog Timer registers

Table 4-2. Memory layout of Watchdog Timer registers

Offset	+3	+2	+1	+0
0x00000000	WDG_PROT 00000000 00000000 00000000 00000000			
0x00000004	reserved 00000000 00000000 00000000 00000000			
0x00000008	WDG_CNT 00000000 00000000 00000000 00000000			
0x0000000C	WDG_RSTCAUSE 00000000 00000000 00000000 000XXXXX			
0x00000010	WDG_TRG0 00000000 00000000 00000000 00000000			
0x00000014	reserved 00000000 00000000 00000000 00000000			
0x00000018	WDG_TRG1 00000000 00000000 00000000 00000000			
0x0000001C	reserved 00000000 00000000 00000000 00000000			
0x00000020	WDG_INT 00000000 00000000 00000000 00000000			
0x00000024	WDG_INTCLR 00000000 00000000 00000000 00000000			
0x00000028	reserved 00000000 00000000 00000000 00000000			
0x0000002C	WDG_TRG0CFG 00000000 00000000 00000000 00000000			
0x00000030	WDG_TRG1CFG 00000000 00000000 00000000 00000000			
0x00000034	WDG_RUNLL 00000000 00000000 00000000 00000000			
0x00000038	WDG_RUNUL 00000001 00000000 00000000 00000000			

Table 4-2. Memory layout of Watchdog Timer registers

Offset	+3	+2	+1	+0
0x0000003C	WDG_PSSLL 00000000 00000000 00000000 00000000			
0x00000040	WDG_PSSUL 10000000 00000000 00000000 00000000			
0x00000044	WDG_RSTDLY 00000000 00000000 00000000 00000000			
0x00000048	WDG_CFG 00000000 00000000 00000000 00000011			

- 'S' implies access with protected sequence.
- 'X' implies that the read value cannot be determined.

4.2.1 Watchdog Protection Register (WDG_PROT)

The Watchdog Protection Register (WDG_PROT) must be written with the correct key, i.e. 0xEDACCE55, to unlock write access to other Watchdog registers except the trigger registers, which are used to clear the Watchdog counter. The registers are locked again after the next write to the configuration registers. WDG_PROT should be written to using 32-bit writes only. The recommended sequence of events involves first writing to the WDG_PROT register to enable write access followed by a write access to the Watchdog configuration register. Writing to this register is also possible after the WDG_CFG:LOCK bit has been set.

Watchdog Protection Register (WDG_PROT)

Figure 4-2. Watchdog Protection Register (WDG_PROT)

WDG_PROT																															
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**
KEY[31]	KEY[30]	KEY[29]	KEY[28]	KEY[27]	KEY[26]	KEY[25]	KEY[24]	KEY[23]	KEY[22]	KEY[21]	KEY[20]	KEY[19]	KEY[18]	KEY[17]	KEY[16]	KEY[15]	KEY[14]	KEY[13]	KEY[12]	KEY[11]	KEY[10]	KEY[9]	KEY[8]	KEY[7]	KEY[6]	KEY[5]	KEY[4]	KEY[3]	KEY[2]	KEY[1]	KEY[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-3. Watchdog Protection Register (WDG_PROT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	KEY	Protection bits The key 0xEDACCE55 unlocks the Watchdog registers for writing. Writing any other value to this register causes a data abort. When Watchdog registers are unlocked, reading this register returns 0xFFFFFFFF. When Watchdog registers are locked, reading this register returns 0x00000000.

Note: An incorrect key, writing to a Watchdog register without first unlocking it or writing twice to the Watchdog Protection Register causes a data abort. For more details on Watchdog register configuration refer to 4.4 Notes on using the Watchdog Timer and the paragraph titled 'Register configuration'.

4.2.2 Watchdog Counter Register (WDG_CNT)

The Watchdog Counter Register (WDG_CNT) shows the current counting value of the Watchdog counter.

Watchdog Counter Register (WDG_CNT)

Figure 4-3. Watchdog Counter Register (WDG_CNT)

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

Table 4-4. Watchdog Counter Register (WDG_CNT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	WDGCNT	Watchdog Counter These bits show the current Watchdog counter value. The Watchdog counter value is sampled with the bus clock.

Note. :

The Watchdog counter register lags the Watchdog counter by a certain range because of sampling and clock synchronization delays. This relationship is expressed by:

$$\text{Watchdog counter register} = \text{Watchdog counter} - (1\text{to}4) - \frac{(2\text{to}3) \times \text{Wachdogclockfrequency}}{\text{AHBBusclockfrequency}}$$

The uncertainty in the above equation is due to the asynchronous relationship between the Watchdog clock and bus clock.

4.2.3 Watchdog Reset Cause Register (WDG_RSTCAUSE)

The Watchdog Reset Cause Register (WDG_RSTCAUSE) is a status register that, when read, shows the cause of a Watchdog reset/NMI from the Watchdog module. This register is not initialised on any reset. For more details for the cause of a Watchdog reset refer to Watchdog error in 4.4 Notes on using the Watchdog Timer.

Watchdog Reset Cause Register (WDG_RSTCAUSE)

Figure 4-4. Watchdog Reset Cause Register (WDG_RSTCAUSE)

WDG_RSTCAUSE																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RSTCAUSE4	RSTCAUSE3	RSTCAUSE2	RSTCAUSE1	RSTCAUSE0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RW0PS	RW0PS	RW0PS	RW0PS	RW0PS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X

Table 4-5. Watchdog Reset Cause Register (WDG_RSTCAUSE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4]	RSTCAUSE4	Reset Cause bit 4 This bit is set to '1' when the Watchdog counter is triggered while the LOCK bit in the Watchdog Configuration Register (i.e. WDG_CFG:LOCK) is still '0'. '0': Bit is cleared '1': No effect

Table 4-5. Watchdog Reset Cause Register (WDG_RSTCAUSE) bits

Bit Position	Bit Field Name	Bit Description
[3]	RSTCAUSE3	Reset Cause bit 3 This bit is set to '1' when the Watchdog counter is triggered before the counter crosses the lower limit of the Watchdog window defined in WDG_RUNLL/WDG_PSSLL. '0': Bit is cleared '1': No effect
[2]	RSTCAUSE2	Reset Cause bit 2 This bit is set to '1' when the Watchdog counter reaches the upper limit of the Watchdog window defined in WDG_RUNUL/WDG_PSSUL. '0': Bit is cleared '1': No effect
[1]	RSTCAUSE1	Reset Cause bit 1 This bit is set to '1' when the triggering sequence in the trigger registers has been violated. For more details on the triggering sequence, refer to 4.4 Notes on using the Watchdog Timer) and the paragraph titled 'Triggering the Watchdog'. '0': Bit is cleared '1': No effect
[0]	RSTCAUSE0	Reset Cause bit 0 This bit is set to '1' when the value written to the Trigger Register, either WDG_TRG0 or WDG_TRG1 does not match the configured value in the corresponding Trigger Configuration Register, WDG_TRG0CFG or WDG_TRG1CFG. '0': Bit is cleared '1': No effect

Note: This register must be cleared when configuring the Watchdog. The register values are valid when a Watchdog reset is generated. The Watchdog reset is captured in the System Controller Reset Cause register (SYSC_RSTCAUSEUR) described in 2. System Controller.

4.2.4 Watchdog Trigger 0 Register (WDG_TRG0)

The Watchdog Trigger 0 Register (WDG_TRG0) defines the first stage of the Watchdog trigger sequence needed to clear the Watchdog counter. This register needs to be written with the value defined in the WDG_TRG0CFG register; writing any other value results in a Watchdog reset/NMI. Depending on the PPU setting, this register can be written in non-privileged mode.

Watchdog Trigger 0 Register (WDG_TRG0)

Figure 4-5. Watchdog Trigger 0 Register (WDG_TRG0)

WDG_TRG0																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WDGTRG0[7]	WDGTRG0[6]	WDGTRG0[5]	WDGTRG0[4]	WDGTRG0[3]	WDGTRG0[2]	WDGTRG0[1]	WDGTRG0[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-6. Watchdog Trigger 0 Register (WDG_TRG0) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	WDGTRG0	Watchdog Trigger 0 Register bits As the first stage of the Watchdog trigger sequence, these bits need to be written with the value contained in the Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG). Reading these bits returns '0'.

Note: Figure 4-22 explains the flow of the Watchdog trigger sequence.

4.2.5 Watchdog Trigger 1 Register (WDG_TRG1)

Watchdog Trigger 1 Register (WDG_TRG1) defines the second and final stage of Watchdog trigger sequence needed to clear the Watchdog counter. This register must be written with the value defined in the WDG_TRG1CFG register; writing any other value results in a Watchdog reset/NMI. Depending on the PPU setting, this register can be written in non-privileged mode.

Watchdog Trigger 1 Register (WDG_TRG1)

Figure 4-6. Watchdog Trigger 1 Register (WDG_TRG1)

WDG_TRG1																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WDGTRG1[7]	WDGTRG1[6]	WDGTRG1[5]	WDGTRG1[4]	WDGTRG1[3]	WDGTRG1[2]	WDGTRG1[1]	WDGTRG1[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-7. Watchdog Trigger 1 Register (WDG_TRG1) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	WDGTRG1	Watchdog Trigger 1 Register bits As the final stage of the Watchdog trigger sequence, these bits need to be written with the value contained in the Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG). Reading these bits returns '0'.

4.2.6 Watchdog Interrupt Configuration Register (WDG_INT)

The Watchdog Interrupt Configuration Register (WDG_INT) is used to configure all Watchdog module related interrupt settings. This register also contains the interrupt status flags.

Watchdog Interrupt Configuration Register (WDG_INT)

Figure 4-7. Watchdog Interrupt Configuration Register (WDG_INT)

WDG_INT																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMIEN	IRQEN	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMIFLAG	IRQFLAG
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-8. Watchdog Interrupt Configuration Register (WDG_INT) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:18]	read0	-
[17]	NMIEN	Reset/NMI Configuration bit This bit generates either a Watchdog reset or an NMI from the Watchdog module when there is a Watchdog error condition. '0': Generate a Watchdog reset on a Watchdog error '1': Generate an NMI on a Watchdog error **Note:** This bit should only be used as a debug/test feature. If it is set to '1' in a real application, then safety is greatly violated. This bit has three times redundancy; majority vote function and automatic correction is implemented in the design to take care of the bit flip scenario.
[16]	IRQEN	Prewarn Interrupt Enable bit This bit enables the generation of a prewarn interrupt. Reset generation is delayed if the Watchdog Reset Delay Counter Register (i.e. WDG_RSTDLY) is programmed with a value other than 0x0000 on a Watchdog error. '0': Prewarn interrupt is disabled '1': Prewarn interrupt is enabled

Table 4-8. Watchdog Interrupt Configuration Register (WDG_INT) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:2]	read0	-
[1]	NMIFLAG	NMI Flag This bit is set when any Watchdog error condition occurs and the Reset/NMI Configuration bit (i.e. WDG_INT:NMIEN) is '1'. If WDG_INT:NMIEN is '0', a reset is generated. '0': No Watchdog error condition detected '1': Watchdog error condition detected This bit is cleared by writing '1' to the Non-Maskable Interrupt (NMI) Clear bit in the Watchdog Interrupt Clear Register (i.e. WDG_INTCLR:NMICLR).
[0]	IRQFLAG	Prewarn Interrupt Flag This bit is set when any Watchdog error condition occurs. A prewarn interrupt is generated when the Prewarn Interrupt Enable bit in this register (i.e. WDG_INT:IRQEN) is '1'. '0': No Watchdog error condition detected '1': Watchdog error condition detected This bit is cleared by writing '1' to the Prewarn Interrupt Clear bit in the Watchdog Interrupt Clear Register (i.e. WDG_INTCLR:IRQCLR).

Notes:

1. For more information about reset/prewarn interrupt/NMI refer to 'Generation of prewarn interrupt and reset/NMI'.
2. For more details about the Watchdog error condition, refer to 'Watchdog error'.

4.2.7 Watchdog Interrupt Clear Register (WDG_INTCLR)

The Watchdog Interrupt Clear Register (WDG_INTCLR) clears the NMI flag and prewarn interrupt flag. Writing to this register is also possible after the WDG_CFG:LOCK bit is set.

Watchdog Interrupt Clear Register (WDG_INTCLR)

Figure 4-8. Watchdog Interrupt Clear Register (WDG_INTCLR)

WDG_INTCLR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMICLR	IRQCLR
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0W1PS	R0W1PS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-9. Watchdog Interrupt Clear Register (WDG_INTCLR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	NMICLR	NMI Clear bit This bit clears the NMI flag WDG_INT:NMIFLAG. '0': No effect '1': Clears the NMI flag
[0]	IRQCLR	Prewarn Interrupt Clear bit This bit clears the prewarn interrupt flag WDG_INT:IRQFLAG. '0': No effect '1': Clears the prewarn interrupt flag

4.2.8 Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG)

The Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG) defines the valid value for the WDG_TRG0 register. This value is needed for the Watchdog trigger sequence.

Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG)

Figure 4-9. Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG)

WDG_TRG0CFG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WDGTRG0CFG[7]	WDGTRG0CFG[6]	WDGTRG0CFG[5]	WDGTRG0CFG[4]	WDGTRG0CFG[3]	WDGTRG0CFG[2]	WDGTRG0CFG[1]	WDGTRG0CFG[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-10. Watchdog Trigger 0 Configuration Register (WDG_TRG0CFG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	WDGTRG0CFG	Watchdog Trigger 0 Register Configuration bits These bits define the value to be written to the Watchdog Trigger 0 Register (WDG_TRG0) as the first part of the trigger sequence. In case the Watchdog should be cleared, this value has to be written to the WDG_TRG0 register as the first part of the trigger sequence.

4.2.9 Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG)

The Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG) defines the valid value for the WDG_TRG1 register. This value is needed for the Watchdog trigger sequence.

Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG)

Figure 4-10. Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG)

WDG_TRG1CFG																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WDGTRG1CFG[7]	WDGTRG1CFG[6]	WDGTRG1CFG[5]	WDGTRG1CFG[4]	WDGTRG1CFG[3]	WDGTRG1CFG[2]	WDGTRG1CFG[1]	WDGTRG1CFG[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-11. Watchdog Trigger 1 Configuration Register (WDG_TRG1CFG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	WDGTRG1CFG	Watchdog Trigger 1 Register Configuration bits These bits define the value to be written to the Watchdog Trigger 1 Register (WDG_TRG1) as the final part of the trigger sequence. In case the Watchdog should be cleared, this value has to be written to the WDG_TRG1 register as the second part of the trigger sequence.

4.2.10 Watchdog Lower Limit RUN Register (WDG_RUNLL)

The Watchdog Lower Limit RUN Register (WDG_RUNLL) contains a value which defines the lower border of the Watchdog window when the device is in the RUN state and before WDG_CFG:LOCK is set. After WDG_CFG:LOCK is set, the value written to this register is used for the operation of the Watchdog counter. If the device state is changed from PSS to RUN, the Watchdog uses this register value only after the first Watchdog trigger.

Watchdog Lower Limit RUN Register (WDG_RUNLL)

Figure 4-11. Watchdog Lower Limit RUN Register (WDG_RUNLL)

WDG_RUNLL																															
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**
WDGRUNLL[31]	WDGRUNLL[30]	WDGRUNLL[29]	WDGRUNLL[28]	WDGRUNLL[27]	WDGRUNLL[26]	WDGRUNLL[25]	WDGRUNLL[24]	WDGRUNLL[23]	WDGRUNLL[22]	WDGRUNLL[21]	WDGRUNLL[20]	WDGRUNLL[19]	WDGRUNLL[18]	WDGRUNLL[17]	WDGRUNLL[16]	WDGRUNLL[15]	WDGRUNLL[14]	WDGRUNLL[13]	WDGRUNLL[12]	WDGRUNLL[11]	WDGRUNLL[10]	WDGRUNLL[9]	WDGRUNLL[8]	WDGRUNLL[7]	WDGRUNLL[6]	WDGRUNLL[5]	WDGRUNLL[4]	WDGRUNLL[3]	WDGRUNLL[2]	WDGRUNLL[1]	WDGRUNLL[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-12. Watchdog Lower Limit RUN Register (WDG_RUNLL) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	WDGRUNLL	Lower Border of Watchdog Window in RUN State These bits define the lower limit of the Watchdog window in the RUN state. Reading these bits always reflects the written value independent of the value of the LOCK bit in the Configuration Register (WDG_CFG:LOCK).

4.2.11 Watchdog Upper Limit RUN Register (WDG_RUNUL)

The Watchdog Upper Limit RUN Register (WDG_RUNUL) contains a value which defines the upper border of the Watchdog window when the device is in the RUN state and before WDG_CFG:LOCK is set. After WDG_CFG:LOCK is set the value written to this register is used for the operation of the Watchdog Counter. If the device state is changed from PSS to RUN, the Watchdog uses this register value only after the first Watchdog trigger.

Watchdog Upper Limit RUN Register (WDG_RUNUL)

Figure 4-12. Watchdog Upper Limit RUN Register (WDG_RUNUL)

WDG_RUNUL																															
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**
WDGRUNUL[31]	WDGRUNUL[30]	WDGRUNUL[29]	WDGRUNUL[28]	WDGRUNUL[27]	WDGRUNUL[26]	WDGRUNUL[25]	WDGRUNUL[24]	WDGRUNUL[23]	WDGRUNUL[22]	WDGRUNUL[21]	WDGRUNUL[20]	WDGRUNUL[19]	WDGRUNUL[18]	WDGRUNUL[17]	WDGRUNUL[16]	WDGRUNUL[15]	WDGRUNUL[14]	WDGRUNUL[13]	WDGRUNUL[12]	WDGRUNUL[11]	WDGRUNUL[10]	WDGRUNUL[9]	WDGRUNUL[8]	WDGRUNUL[7]	WDGRUNUL[6]	WDGRUNUL[5]	WDGRUNUL[4]	WDGRUNUL[3]	WDGRUNUL[2]	WDGRUNUL[1]	WDGRUNUL[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-13. Watchdog Upper Limit RUN Register (WDG_RUNUL) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	WDGRUNUL	Upper Border of Watchdog Window in RUN State These bits define the upper limit of the Watchdog window in the RUN state. Reading these bits always reflects the written value independent of the value of the LOCK bit in the Configuration Register (WDG_CFG:LOCK).

4.2.12 Watchdog Lower Limit PSS Register (WDG_PSSLL)

The Watchdog Lower Limit PSS Register (WDG_PSSLL) contains a value that defines the lower border of the Watchdog window when the device is in PSS; the Watchdog is configured to also run during PSS. The configured value in the register is used for the operation of the Watchdog Counter only after WDG_CFG:LOCK is set. After the device changes state from PSS to RUN, this register value is used for Watchdog window operation until the first time clear of the Watchdog counter.

Watchdog Lower Limit PSS Register (WDG_PSSLL)

Figure 4-13. Watchdog Lower Limit PSS Register (WDG_PSSLL)

WDG_PSSLL																															
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**
WDGPSSLL[31]	WDGPSSLL[30]	WDGPSSLL[29]	WDGPSSLL[28]	WDGPSSLL[27]	WDGPSSLL[26]	WDGPSSLL[25]	WDGPSSLL[24]	WDGPSSLL[23]	WDGPSSLL[22]	WDGPSSLL[21]	WDGPSSLL[20]	WDGPSSLL[19]	WDGPSSLL[18]	WDGPSSLL[17]	WDGPSSLL[16]	WDGPSSLL[15]	WDGPSSLL[14]	WDGPSSLL[13]	WDGPSSLL[12]	WDGPSSLL[11]	WDGPSSLL[10]	WDGPSSLL[9]	WDGPSSLL[8]	WDGPSSLL[7]	WDGPSSLL[6]	WDGPSSLL[5]	WDGPSSLL[4]	WDGPSSLL[3]	WDGPSSLL[2]	WDGPSSLL[1]	WDGPSSLL[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-14. Watchdog Lower Limit PSS Register (WDG_PSSLL) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	WDGPSSLL	Lower Border of Watchdog Window in PSS State These bits define the lower limit of the Watchdog window in Power Saving State (PSS). Reading these bits always reflects the written value independent of the value of the LOCK bit in the Configuration Register (WDG_CFG:LOCK).

4.2.13 Watchdog Upper Limit PSS Register (WDG_PSSUL)

The Watchdog Upper Limit PSS Register (WDG_PSSUL) contains a value which defines the upper border of the Watchdog window when the device is in PSS; the Watchdog is configured to also run during PSS. The configured value of the register is used for the operation of the Watchdog Counter only after WDG_CFG:LOCK is set. After the device changes state from PSS to RUN, this register value is used for Watchdog window operation until the first time clear Watchdog counter.

Watchdog Upper Limit PSS Register (WDG_PSSUL)

Figure 4-14. Watchdog Upper Limit PSS Register (WDG_PSSUL)

WDG_PSSUL																															
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**
WDGPSSUL[31]	WDGPSSUL[30]	WDGPSSUL[29]	WDGPSSUL[28]	WDGPSSUL[27]	WDGPSSUL[26]	WDGPSSUL[25]	WDGPSSUL[24]	WDGPSSUL[23]	WDGPSSUL[22]	WDGPSSUL[21]	WDGPSSUL[20]	WDGPSSUL[19]	WDGPSSUL[18]	WDGPSSUL[17]	WDGPSSUL[16]	WDGPSSUL[15]	WDGPSSUL[14]	WDGPSSUL[13]	WDGPSSUL[12]	WDGPSSUL[11]	WDGPSSUL[10]	WDGPSSUL[9]	WDGPSSUL[8]	WDGPSSUL[7]	WDGPSSUL[6]	WDGPSSUL[5]	WDGPSSUL[4]	WDGPSSUL[3]	WDGPSSUL[2]	WDGPSSUL[1]	WDGPSSUL[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-15. Watchdog Upper Limit PSS Register (WDG_PSSUL) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	WDGPSSUL	Upper Border of Watchdog Window in PSS State These bits define the upper limit of the Watchdog window in PSS. Reading these bits always reflects the written value independent of the value of the LOCK bit in the Configuration Register (WDG_CFG:LOCK).

Note: The PSS limit defined by the WDG_PSSUL and WDG_PSSLL registers is used by the Watchdog from the time CPU enters standby mode (device PSS state) until the first Watchdog trigger after the device wakeup.

4.2.14 Watchdog Reset Delay Counter Register (WDG_RSTDLY)

The Watchdog Reset Delay Counter Register (WDG_RSTDLY) is used to delay the Watchdog reset/NMI in case of a Watchdog error. When a Watchdog error occurs, the WDG_INT:IRQFLAG is set and a Prewarn interrupt is generated if the WDG_INT:IRQEN bit is set. After WDG_INT:IRQFLAG has been set, this register counts down to zero using the count clock of the Watchdog. Upon reaching zero, a Watchdog reset or NMI is generated depending on the WDG_INT:NMIEN bit settings. If the value already contained in this register is zero, a reset/NMI is generated immediately upon Watchdog error.

Watchdog Reset Delay Counter Register (WDG_RSTDLY)

Figure 4-15. Watchdog Reset Delay Counter Register (WDG_RSTDLY)

WDG_RSTDLY																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WDGRSTDLY[15]	WDGRSTDLY[14]	WDGRSTDLY[13]	WDGRSTDLY[12]	WDGRSTDLY[11]	WDGRSTDLY[10]	WDGRSTDLY[9]	WDGRSTDLY[8]	WDGRSTDLY[7]	WDGRSTDLY[6]	WDGRSTDLY[5]	WDGRSTDLY[4]	WDGRSTDLY[3]	WDGRSTDLY[2]	WDGRSTDLY[1]	WDGRSTDLY[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS	R0WPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-16. Watchdog Reset Delay Counter Register (WDG_RSTDLY) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	WDGRSTDLY	Delay to Reset/NMI These bits define the delay to be inserted in Watchdog reset/NMI generation in case of a Watchdog error. The value in this register is decremented to '0' with the Watchdog clock, upon which a reset/NMI is generated.

Note: Due to the synchronisation delay, two more Watchdog cycles are consumed before the delay counter is triggered to count down when a Watchdog error occurs. This results in an actual delay of (WDG_RSTDLY + 2) Watchdog clock cycles before a reset/NMI is generated when the value in the WDG_RSTDLY register is not zero. If the value in the register is zero, then a reset/NMI is generated immediately upon Watchdog error.

4.2.15 Watchdog Configuration Register (WDG_CFG)

The Watchdog Configuration Register (WDG_CFG) is used to configure the operation of the Watchdog Timer.

Watchdog Configuration Register (WDG_CFG)

Figure 4-16. Watchdog Configuration Register (WDG_CFG)

WDG_CFG																													
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**
read0	read0	read0	read0	read0	read0	read0	LOCK	read0	read0	read0	OBSSEL[4]	OBSSEL[3]	OBSSEL[2]	OBSSEL[1]	OBSSEL[0]	read0	read0	read0	read0	read0	read0	CLKSEL[1]	CLKSEL[0]	read0	read0	read0	read0	DBGEN	ALLOWSTOPCLK	WDENPSS	WDENRUN
R0	R0	R0	R0	R0	R0	R0	RWPS1	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 4-17. Watchdog Configuration Register (WDG_CFG) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	LOCK	Lock bit This bit is written one time only. It is used to prevent reconfiguration of the Watchdog during run time. If this bit is '0', then the Watchdog counter cannot be cleared by the trigger sequence even though configuration registers can be written to. When this bit is set to '1', the configured values of the Watchdog window become valid and the Watchdog counter is automatically cleared. Clearing the Watchdog counter using the trigger sequence is only possible if this bit is set to '1'. '0': Write access to configuration registers is possible '1': Write access to configuration registers is not possible with the exception of the trigger registers This bit has three times redundancy; majority vote function and automatic correction is implemented in the design to take care of bit flip scenarios.
[23:21]	read0	-

Table 4-17. Watchdog Configuration Register (WDG_CFG) bits

Bit Position	Bit Field Name	Bit Description
[20:16]	OBSSEL	Observe bits of Watchdog Counter These bits select a particular bit in the Watchdog counter (WDG_CNT) for observation. '00000':1st bit is selected for output '00001':2nd bit is selected for output .... '11111':32nd bit is selected for output **Note:** For more details on observing outputs, please refer to 4.3.2 Watchdog operation).
[15:10]	read0	-
[9:8]	CLKSEL	Clock Selection bits These bits are used to select the clock for the Watchdog counter '00': RC clock is selected '01': Slow RC clock is selected '10': Sub clock is selected '11': Main clock is selected On start up, the Watchdog counter starts on the RC clock. Switching to the clock configured in the CLKSEL bits happens when the device switches to a new state. For more details refer to 4.4 Notes on using the Watchdog Timer and the paragraph titled 'Watchdog clock configuration' and 2. System Controller in this hardware manual.
[7:4]	read0	-
[3]	DBGEN	Debug Enable This bit enables/disables debug mode for the Watchdog. '0': Debug mode disabled '1': Debug mode enabled When DBGEN is set to '1' and the processor is in debug state, the Watchdog counter is stopped. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[2]	ALLOWSTOPCLK	Configuration to Stop the Clock This bit is used to stop the Watchdog clock when the device is entering deep PSS with all the clocks stopped. '0': Watchdog timer clock cannot be stopped entering deep PSS '1': Watchdog timer clock can be stopped entering deep PSS This bit is valid only when the Watchdog counter is enabled in PSS i.e. the WDG_CFG:WDENPSS bit is set to '1'.

Table 4-17. Watchdog Configuration Register (WDG_CFG) bits

Bit Position	Bit Field Name	Bit Description
[1]	WDENPSS	Enable bit for Watchdog Counter in PSS State. This bit is used to enable the Watchdog counter in PSS. It only becomes valid when the WDG_CFG:LOCK bit is set to '1'. '0': Disable Watchdog counter in PSS state '1': Enable Watchdog counter in PSS state This bit has three times redundancy; majority vote function and automatic correction is implemented in the design to take care of bit flip scenarios.
[0]	WDENRUN	Enable bit for Watchdog Counter in RUN State. This bit is used to enable the Watchdog counter in the RUN state. It only becomes valid when the WDG_CFG:LOCK bit is set to '1'. '0': Disable Watchdog counter in RUN state '1': Enable Watchdog counter in RUN state This bit has three times redundancy; majority vote function and automatic correction is implemented in the design to take care of bit flip scenarios.

Note: If the WDG_CFG:OBSSEL bits are configured after the corresponding pin configuration register is configured to output the Watchdog output, there may be an initial glitch on this pin. However, the output should stabilise after one Watchdog clock cycle.

4.3 Operation of the Watchdog Timer

This section describes the operation of the Watchdog Timer

The Watchdog Timer is used to detect a hang in a user program. If a Watchdog error occurs, then the Watchdog Timer generates corrective action in the form of a reset/NMI.

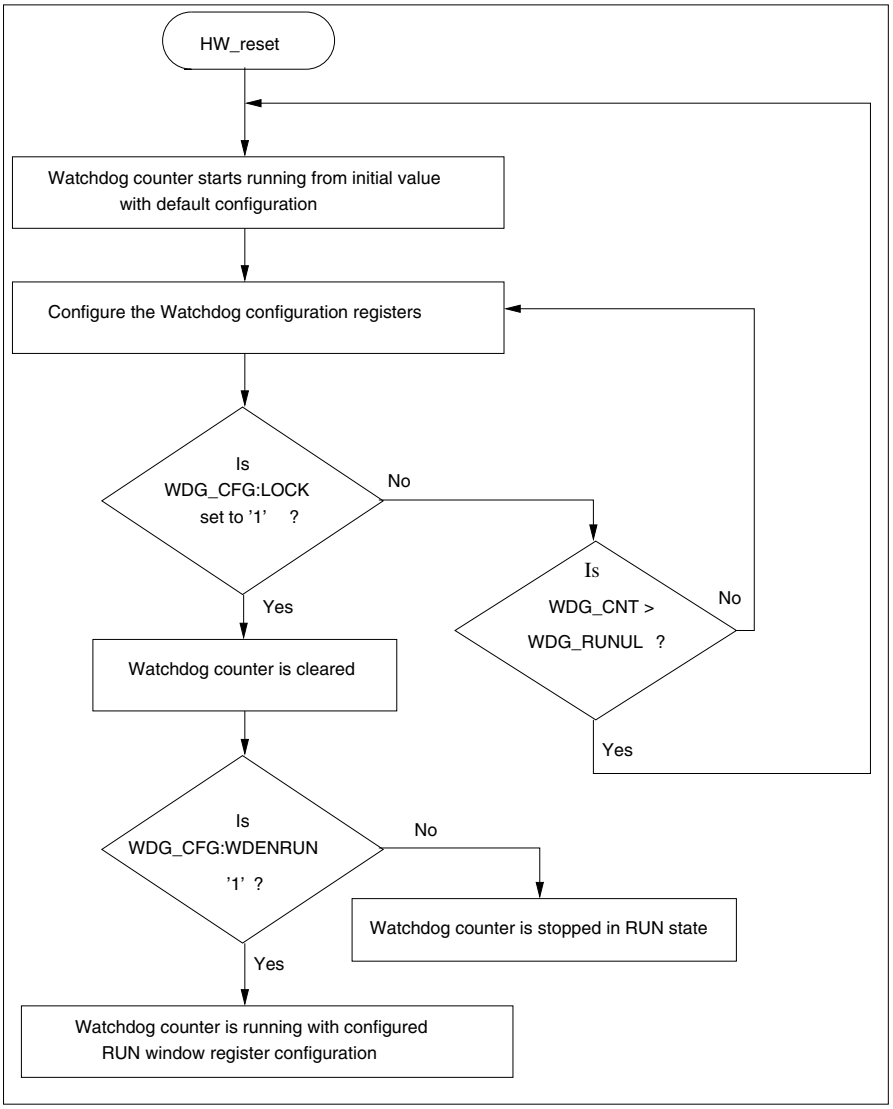
4.3.1 Watchdog startup

The following steps explain the Watchdog startup:

1. The Watchdog Timer is enabled by a hard reset.
2. The Watchdog counter is initialised with a hard reset and starts counting up.
3. The default value of the RUN window registers i.e. WDG_RUNLL and WDG_RUNUL define the Watchdog window size at startup. After reset, the Watchdog window is set to the maximum size (i.e. lower limit = 0x00000000, upper limit = 0x01000000), which translates into a Watchdog window size of 1.04 s (@ RC clock of 16 MHz).
4. The Watchdog needs to be configured within this Watchdog window and the registers need to be locked by configuring the WDG_CFG:LOCK bit. The configuration registers can be written to as long as the WDG_CFG:LOCK bit is '0'.
5. The configured values in the registers (except the Watchdog window registers and the Watchdog counter enable bits) are valid for the operation of the Watchdog if WDG_CFG:LOCK is '0'. In addition, as long as the WDG_CFG:LOCK bit is '0', the Watchdog continues to run with the default RUN state window size, irrespective of the device state, the Watchdog enable bits (WDG_CFG:WDENRUN and WDG_CFG:WDENPSS) and the Watchdog window size (defined in WDG_RUNLL, WDG_RUNUL, WDG_PSSLL, and WDG_PSSUL).
6. The Watchdog reset is generated if the Watchdog configuration is not locked within the default Watchdog window.
7. Locking the Watchdog window within the default Watchdog window clears the Watchdog counter and applies a new configured RUN state Watchdog window.

For more details on Watchdog Startup refer to 4.1 Outline of the Watchdog Timer.

Figure 4-17. Watchdog Timer startup



4.3.2 Watchdog operation

The Watchdog can operate in RUN state or PSS, depending on the device state (refer to 2. System Controller.)

Watchdog operation in RUN state:

1. If the WDG_CFG:LOCK and WDG_CFG:WDENRUN bits are set, then Watchdog operation is restarted with the RUN state Watchdog window. Watchdog operation in RUN state is illustrated in Figure 4-18
2. If the Watchdog Timer is not triggered when the Watchdog counter is in the Watchdog window or on a wrong trigger sequence, either a Watchdog reset or an NMI is generated (depending on the WDG_INT:NMIEN bit). For the Watchdog trigger sequence and Watchdog errors refer to 4.4 Notes on using the Watchdog Timer.

Watchdog operation in PSS state:

1. When the device state changes to PSS, the Watchdog window follows suit. In PSS, the Watchdog window is defined by the configuration registers WDG_PSSLL and WDG_PSSUL. The Watchdog Timer then runs with PSS Watchdog window as long as the Watchdog is enabled in PSS i.e. the WD_CFG:WDENPSS bit is set. Watchdog operation in PSS is illustrated in Figure 4-19
2. When the device state switches from PSS to RUN, the Watchdog state enable bits are switched immediately (i.e. from WD_CFG:WDENPSS to WD_CFG:WDENRUN). However, even though the Watchdog starts running the counter, the Watchdog window continues in PSS until the Watchdog is triggered.

Note: The Watchdog counter can be cleared by a software reset at any time during Watchdog Timer operation.

Generation of prewarn interrupt and reset/NMI

The following steps explain the generation of the prewarn interrupt and reset/NMI:

1. In the event of any Watchdog errors, the prewarn interrupt flag (WDG_INT:IRQFLAG) is set and a prewarn interrupt generated if WDG_INT:IRQEN is set.
2. The Watchdog reset/NMI generation can be delayed by the value defined in the Watchdog reset delay counter register (WDG_RSTDLY). Any new Watchdog errors are ignored after a prewarn interrupt has been generated and until the Watchdog reset/NMI is generated.
3. After the delay configured in the WDG_RSTDLY register expires, the Watchdog reset/NMI is generated. However, if the value in the Watchdog reset delay counter register (WDG_RSTDLY) is zero, the Watchdog reset/NMI is generated without any delay.
4. Once the WDG_RSTDLY counter starts decrementing, the Watchdog counter cannot be cleared by the trigger sequence.

Note:

It is recommended that the following two factors are applied together:

1. If the WDG_INT:IRQEN bit is set, a prewarn interrupt is generated.
2. If WDG_RSTDLY is not equal to zero, the reset/NMI is delayed.

If WDG_INT:NMIEN = '0', then a Watchdog reset is generated; If WDG_INT:NMIEN = '1', an NMI is generated. Please refer to Figure 4-20.

Note: The non-zero value contained in the WDG_RSTDLY register and WDG_CFG:NMIEN = '1' can only be used for one Watchdog error because the start value of the WDG_RSTDLY counter can only be configured once after a hardware reset. Therefore, if a NMI is generated instead of a reset, then after the first NMI, WDG_RSTDLY counter value is '0' and cannot be changed any longer.

Figure 4-18. Watchdog operation (RUN state)

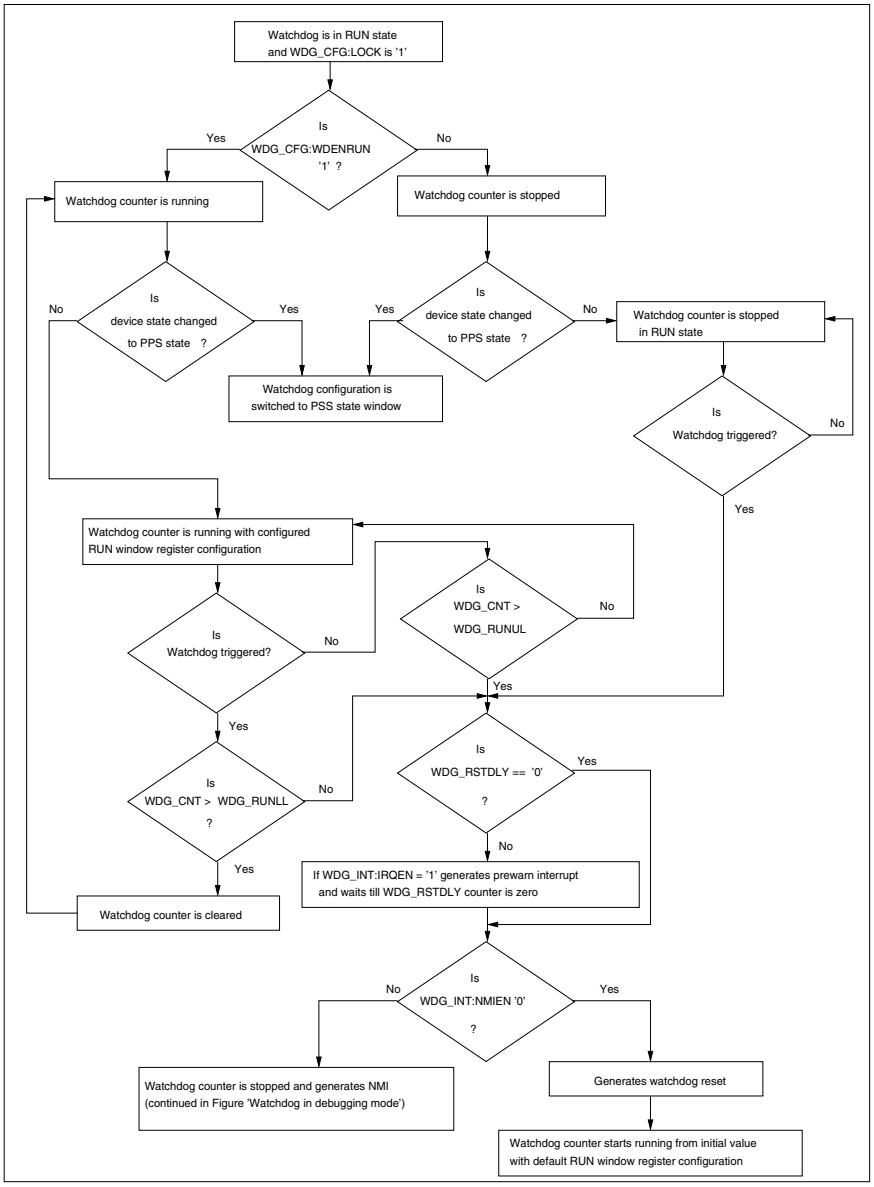


Figure 4-19. Watchdog operation (PSS)

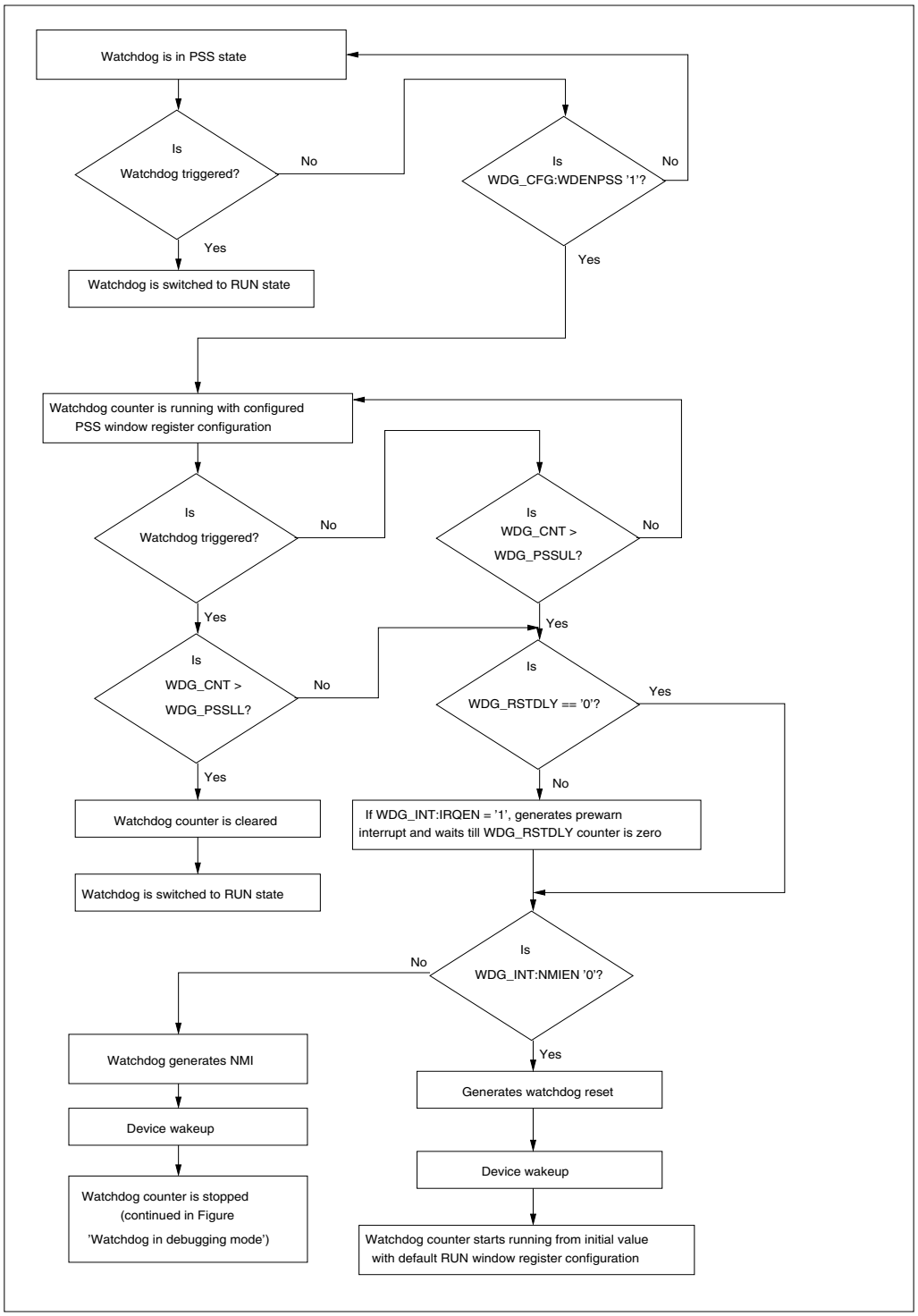
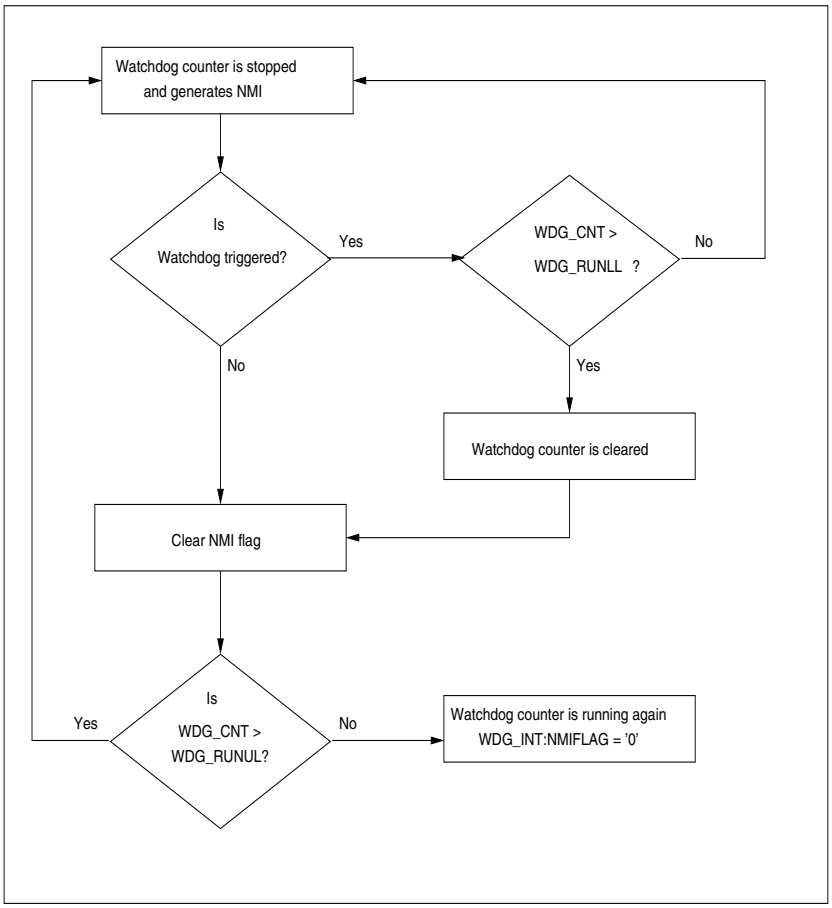


Figure 4-20. Watchdog in debugging mode (WDG_INT:NMIEN = '1')



Selection of the clock source for the Watchdog counter

On startup the Watchdog counter works on the RC clock (CLK_RC). Changing the Watchdog source clock to the Main clock (CLK_MAIN), Sub clock (CLK_SUB) or Slow RC clock (CLK_SRC) is done by configuring the WDG_CFG:CLKSEL bits. However, the new clock setting is delayed until the device switches state to either RUN or PSS. For device state switching refer to the 2. System Controller.

Debug mode

When the WDG_CFG:DBGEN bit is set to '1' and the processor is in debug state, the Watchdog counter stops. When the processor leaves debug state or WDG_CFG:DBGEN is set to '0', the Watchdog resumes operation.

Note: For the definition of debug state, refer to Section 11.8 of the Arm® Cortex® -R4 Technical Reference Manual.

Observe output

The status of the running state of the Watchdog counter can be checked using the WDOG_OBSERVE signal that is output from the MCU. The WDG_CFG:OBSSEL[4:0] bits are used to select any one bit of the Watchdog counter for observation. This signal is valid even when the device is in PSS.

4.4 Notes on using the Watchdog Timer

Essentially the 'Programmer’s guide', this section lists the guidelines for programming the Watchdog module.

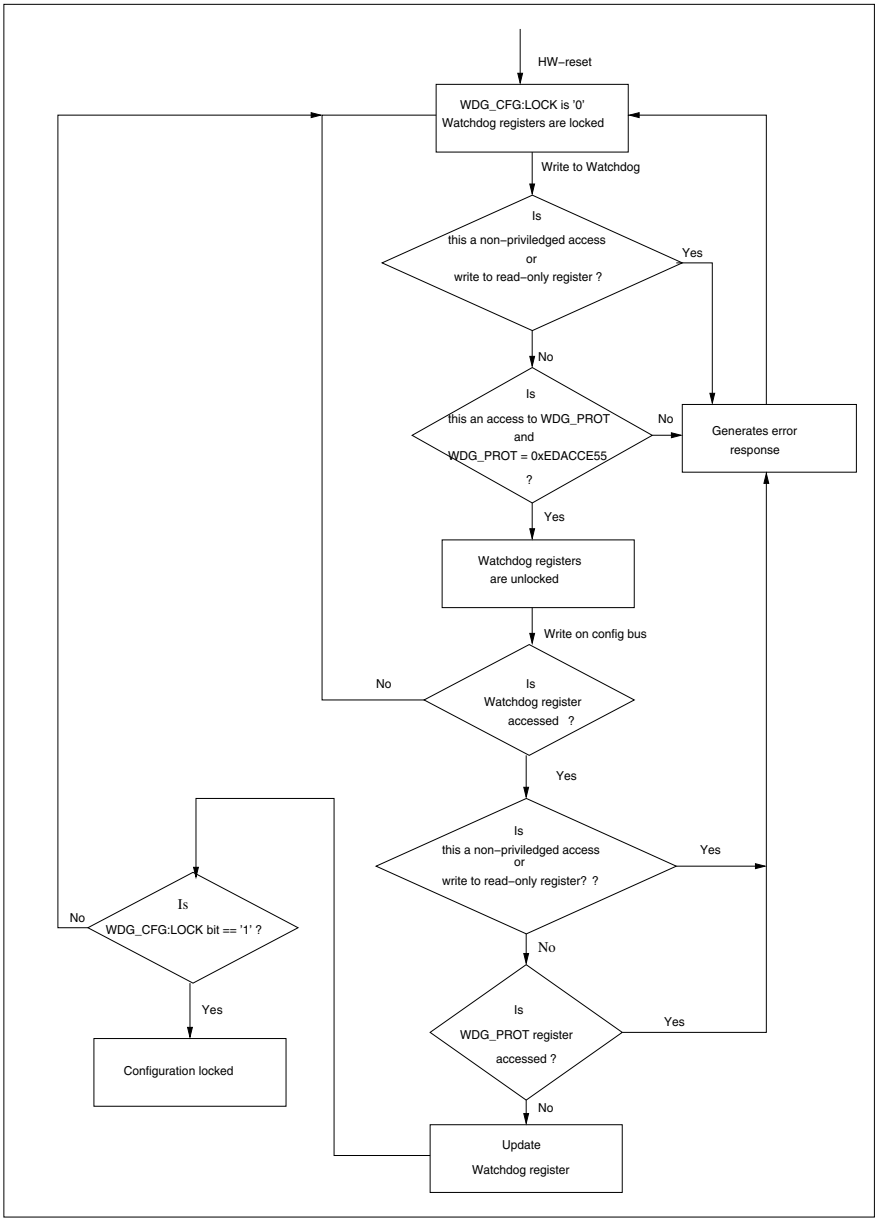
Register configuration

All Watchdog register programming is possible in privileged mode only and when the WDG_CFG:LOCK bit is '0'. In addition, protected sequences are used to protect the Watchdog registers against accidental overwriting. Registers are protected by a predefined 32-bit key which has to be written to a compare key register in order to unlock the Watchdog register set. Write access to any of the protected registers locks the Watchdog register set again and clears the Watchdog Protection Register (WDG_PROT).

If a write access by the same master happens to another register in between the Protection Key Register (WDG_PROT) and the target Watchdog register write access, then a violation of the protection sequence is deemed to have occured.

For more details on the Watchdog register write sequence refer to Figure 4-21.

Figure 4-21. Watchdog register write sequence



Watchdog window configuration

The Watchdog module supports two Watchdog windows i.e. the RUN state window and the PSS window. Which one is used depends on the device state. The user needs to configure the appropriate Watchdog window size for the device state.

Note: If the device is switched from PSS to RUN state, the Watchdog will continue to work within the PSS window until it is triggered, i.e. it switches to the RUN state window only on the first trigger. Watchdog functionality can be enabled or disabled based on the device state by the WDG_CFG:WDENRUN or WDG_CFG:WDENPSS bit setting.

Watchdog clock configuration

The Watchdog clock, together with the Watchdog window configuration register, determines the width of the Watchdog window. As the Watchdog clock state is also part of system profile, Watchdog clock switching is handled by the System Control-

ler during device state switching. For details on the device profile and device state switching, refer to 2. System Controller. Below is the recommended sequence for changing the clock source for the Watchdog counter operation.

1. Enable the new Watchdog clock source by programming the run profile configuration register (SYSC_RUNCKER) in the System Controller, and apply the run trigger by programming the SYSC_TRGRUNCNTR register in the System Controller with 0xAB.
2. Configure the new clock source in the Watchdog by writing to WDG_CFG:CLKSEL and configure the associated Watchdog window by programming the WDG_RUNLL, WDG_RUNUL,WDG_PSSUL, and WDG_PSSLL registers.
3. Apply the Watchdog lock by writing a '1' to the WDG_CFG:LOCK bit.
4. Apply the run trigger again by programming the SYSC_TRGRUNCNTR register in the System Controller with 0xAB. The Watchdog clock is then switched to that selected by the WDG_CFG:CLKSEL bits.

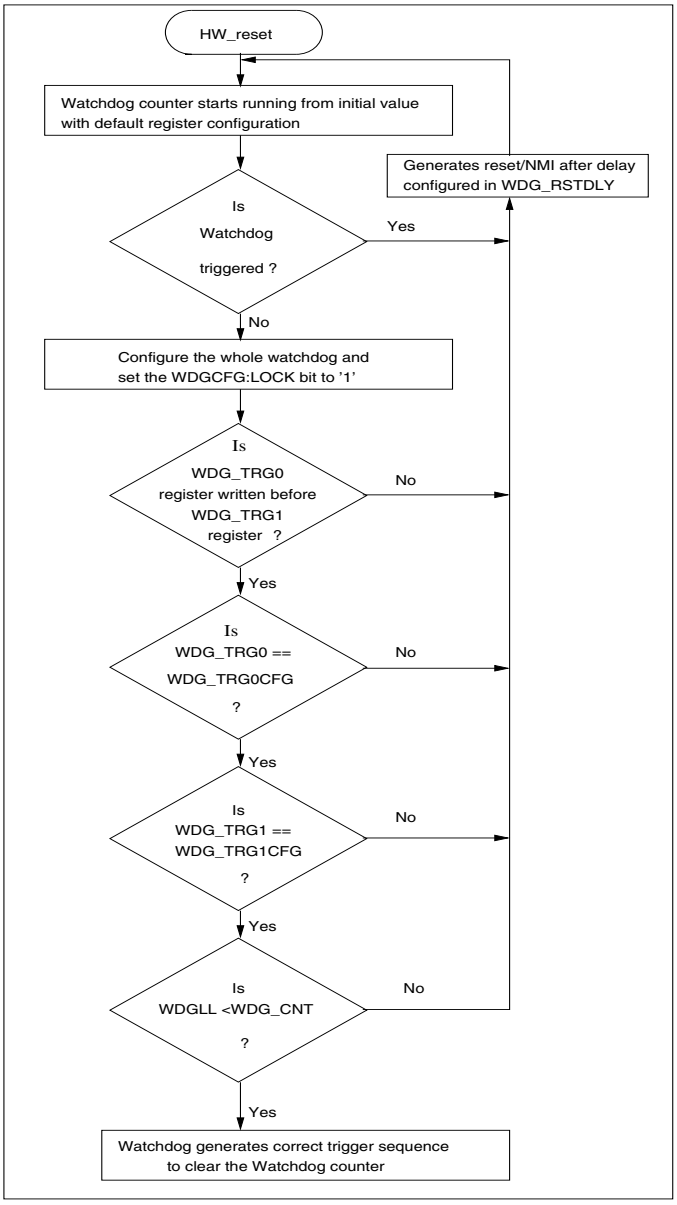
Triggering Watchdog

Triggering the Watchdog clears the Watchdog counter. The trigger sequence involves the following two steps:

1. Program the Watchdog Trigger 0 Register (WDG_TRG0) with the value configured in the WDG_TRG0CFG register.
2. Program the Watchdog Trigger 1 Register (WDG_TRG1) with the value configured in the WDG_TRG1CFG register.

The Watchdog triggering sequence is explained in Figure 4-22; WDGLL can be either WDG_RUNLL or WDG_PSSLL, depending on the device state of the CPU.

Figure 4-22. Triggering sequence of Watchdog



Watchdog error

The following conditions are treated as Watchdog error events and result in the generation of a Watchdog reset/NMI:

- Writing the wrong value to the trigger registers
- Violation of the trigger sequence at the trigger registers (for more details on the trigger sequence refer to 'Triggering Watchdog')
- Not triggering the Watchdog before it reaches the upper limit i.e. the Watchdog counter is greater than or equal to the value configured in Watchdog upper limit register i.e. WDG_RUNUL
- Triggering the Watchdog before the counter has reached the configured lower limit.
- Triggering the Watchdog while the lock bit WDG_CFG:LOCK is still '0'

Causes for bus error response (data abort)

A Bus error response is generated in the following cases:

- A non-existent address is accessed (read or write)
- A write access in non-privileged mode to the configuration register
- A write access to the trigger register in non-privileged mode if it is forbidden by the PPU input
- A violation of a protection sequence of a configuration register (for more details on protection sequence refer to 'Register configuration')
- Writing to a read-only register
- Writing to a configuration register while WDG_CFG:LOCK is '1'
- Writing a wrong value to the WDG_PROT register (to unlock the configuration register for writing)

5. Security Checker

This chapter explains the functions and operations of the Security Checker (SECURITY_CHECKER).

5.1 Outline of Security Checker

This section describes the features and the block diagram of the Security Checker module.

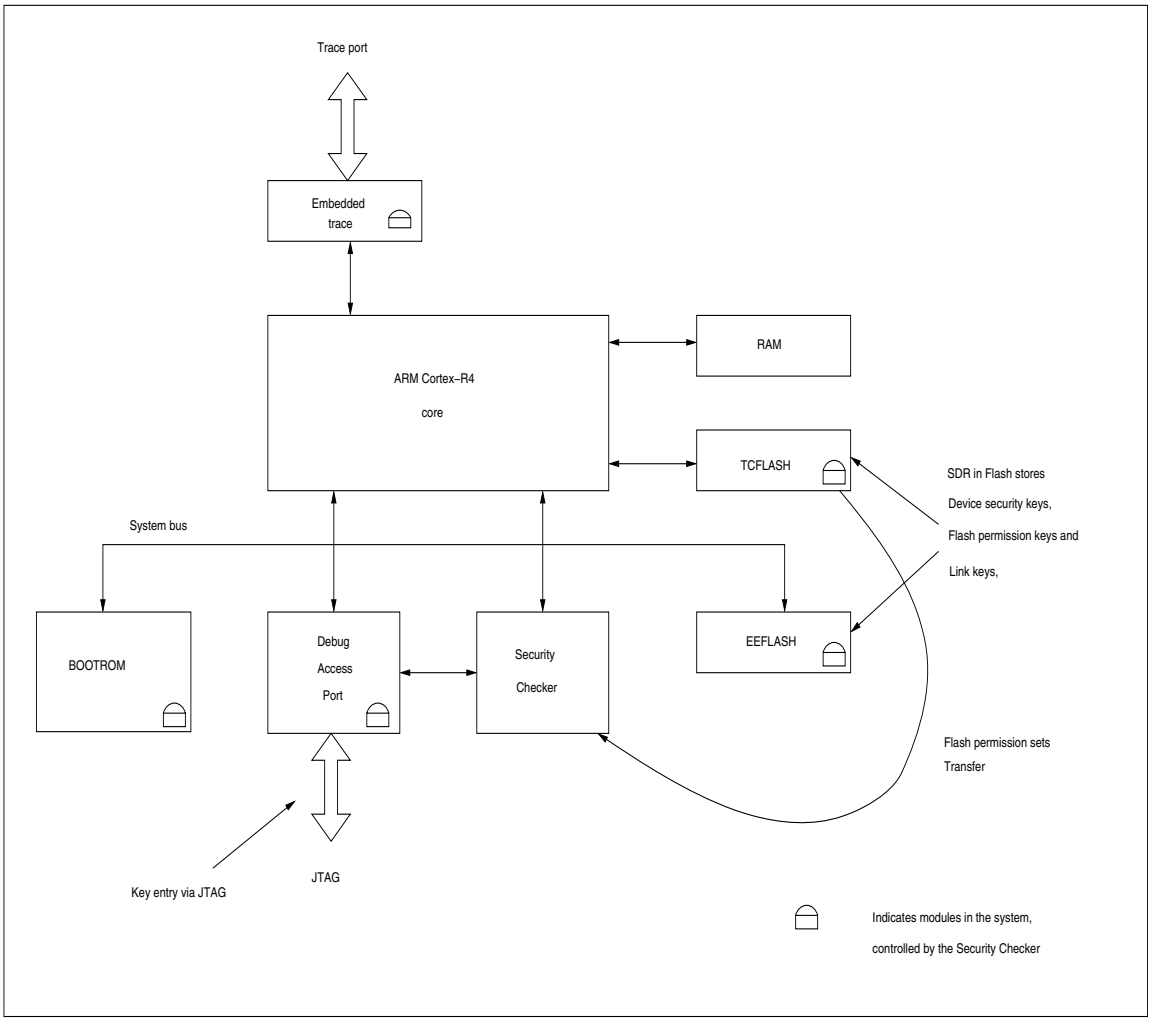
Features of the Security Checker

The Security Checker module controls debug, trace, test, and Flash memory access in the device. It acts as an interface to enter the device security keys and also gives the status of different security settings in the device. The Flash security controls sectorwise access permissions of TCFLASH and EEPROM Flash of the system. It prevents from unauthorized readout of the Flash content. The following section lists the features of Security Checker.

- Supports 128-bit key to prevent unauthorized access to debug, trace, and test modes
- Supports a second 128-bit key for debug, trace, and test access without access to Flash memories e.g. for Field Failure Analysis
- Supports key entry via JTAG interface
- Blocks access to the device when a wrong key is entered until next hard reset
- Supports two 32-bit general purpose registers for exchange of data between debugger and the device even in case of secured devices
- Supports a lock/unlock based safety mechanism for access to Security Checker configuration registers
- Supports selective blocking of JTAG access
- Supports two sets of Flash access permissions and a 128-bit key to switch between these two sets
- Prevents unauthorized readout of Flash content based on link key comparison result
- Provides the status of security of various components in the system

Block diagram of Security Checker

Figure 5-1. Block diagram of Security Checker



5.2 Security Checker registers

This section describes the registers of the Security Checker in detail.

Registers of Security Checker

The following registers are available for Security Checker:

- Flash Security registers
  - TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0~3)
  - EEFLASH Permission User Key Register (SCCFG_EEFPUSRKEY0~3)
- System Interface registers
  - Security Control Register (SCCFG_CTRL)
  - Security Status Register 0 (SCCFG_STAT0)
  - Security Status Register 1 (SCCFG_STAT1)
  - Security Status Register 2 (SCCFG_STAT2)
- Device Security registers
  - Security Key Register (SCCFG_SECKEY0~3)
- General registers
  - Module ID Register (SCCFG_MODID)
  - Security Checker Unlock Register (SCCFG_UNLCK)
  - General Purpose Register (SCCFG_GPREG0~1)

Memory layout of Security Checker registers

Table 5-1. Memory layout of Security Checker registers

Offset	+3	+2	+1	+0
0x00000000 - 0x0000011C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000120	SCCFG_TCFPUSRKEY0 00000000 00000000 00000000 00000000			
0x00000124	SCCFG_TCFPUSRKEY1 00000000 00000000 00000000 00000000			
0x00000128	SCCFG_TCFPUSRKEY2 00000000 00000000 00000000 00000000			
0x0000012C	SCCFG_TCFPUSRKEY3 00000000 00000000 00000000 00000000			
0x00000130	SCCFG_EEFPUSRKEY0 00000000 00000000 00000000 00000000			
0x00000134	SCCFG_EEFPUSRKEY1 00000000 00000000 00000000 00000000			
0x00000138	SCCFG_EEFPUSRKEY2 00000000 00000000 00000000 00000000			
0x0000013C	SCCFG_EEFPUSRKEY3 00000000 00000000 00000000 00000000			
0x00000140 - 0x0000016C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000170	SCCFG_CTRL 00000000 00000000 00000000 00000000			
0x00000174	SCCFG_reserved_53 00000000 00000000 00000000 00000000			
0x00000178	SCCFG_STAT0 00000000 00000000 00000000 00000000			
0x0000017C	SCCFG_STAT1 00000001 0000000X 00000000 00111011			
0x00000180	SCCFG_STAT2 00000000 00000000 00000000 00000101			
0x00000184 - 0x0000018C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000190	SCCFG_SECKEY0 00000000 00000000 00000000 00000000			
0x00000194	SCCFG_SECKEY1 00000000 00000000 00000000 00000000			

Table 5-1. Memory layout of Security Checker registers

Offset	+3	+2	+1	+0
0x00000198	SCCFG_SECKEY2 00000000 00000000 00000000 00000000			
0x0000019C	SCCFG_SECKEY3 00000000 00000000 00000000 00000000			
0x000001A0	SCCFG_MODID 00000000 00000000 00000000 00000000			
0x000001A4	SCCFG_UNLCK 00000000 00000000 00000000 00000000			
0x000001A8	SCCFG_GPREG0 00000000 00000000 00000000 00000000			
0x000001AC	SCCFG_GPREG1 00000000 00000000 00000000 00000000			

Table 5-2. Registers accessible via DAP and system bus

Register	DAP	System bus
SCCFG_CTRL		x
SCCFG_STAT0		x
SCCFG_STAT1		x
SCCFG_STAT2	x	x
SCCFG_MODID	x	x
SCCFG_UNLCK	x	x
SCCFG_GPREG	x	x
SCCFG_SECKEY	x	x
SCCFG_TCFPUSRKEY		x
SCCFG_EEFPUSRKEY		x

Notes:

1. 'x' indicates registers that are accessible.
2. All the registers are accessible by 32-bit access only.

5.2.1 Flash Security registers

5.2.1.1 TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0~3)

A 128-bit wide TCFLASH Permission User Key Register stores the user entered permission key for the TCFLASH. A read access to this register returns zero.

TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0~3)

Figure 5-2. TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0)

SCCFG_TCFPUSRKEY0																															
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**
SCCFGTCFPUSRKEY0[31]	SCCFGTCFPUSRKEY0[30]	SCCFGTCFPUSRKEY0[29]	SCCFGTCFPUSRKEY0[28]	SCCFGTCFPUSRKEY0[27]	SCCFGTCFPUSRKEY0[26]	SCCFGTCFPUSRKEY0[25]	SCCFGTCFPUSRKEY0[24]	SCCFGTCFPUSRKEY0[23]	SCCFGTCFPUSRKEY0[22]	SCCFGTCFPUSRKEY0[21]	SCCFGTCFPUSRKEY0[20]	SCCFGTCFPUSRKEY0[19]	SCCFGTCFPUSRKEY0[18]	SCCFGTCFPUSRKEY0[17]	SCCFGTCFPUSRKEY0[16]	SCCFGTCFPUSRKEY0[15]	SCCFGTCFPUSRKEY0[14]	SCCFGTCFPUSRKEY0[13]	SCCFGTCFPUSRKEY0[12]	SCCFGTCFPUSRKEY0[11]	SCCFGTCFPUSRKEY0[10]	SCCFGTCFPUSRKEY0[9]	SCCFGTCFPUSRKEY0[8]	SCCFGTCFPUSRKEY0[7]	SCCFGTCFPUSRKEY0[6]	SCCFGTCFPUSRKEY0[5]	SCCFGTCFPUSRKEY0[4]	SCCFGTCFPUSRKEY0[3]	SCCFGTCFPUSRKEY0[2]	SCCFGTCFPUSRKEY0[1]	SCCFGTCFPUSRKEY0[0]
R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-3. TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SCCFGTCFPUSRKEY0	TCFLASH User Permission Key If the [126:0] bits of user entered TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY[126:0]) matches with the [126:0] bits of Nominal Flash Permission Key (SCCFG_TCFPSYS-KEY[126:0]) then SCCFG_TCFPUSRKEY[127] bit value determines the TCFLASH permission set. SCCFG_TCFPUSRKEY[127] is '0': Primary permission set is activated (default) '1': Secondary permission set is activated

Note:

Four 32-bit registers are present SCCFG_TCFPUSRKEY0, SCCFG_TCFPUSRKEY1, SCCFG_TCFPUSRKEY2, and SCCF-G_TCFPUSRKEY3 to accommodate 128-bit user entered TCFLASH permission key.

SCCFG_TCFPUSRKEY0 consists of user entered TCFLASH permission key bytes 0 to 3.

SCCFG_TCFPUSRKEY1 consists of user entered TCFLASH permission key bytes 4 to 7.

SCCFG_TCFPUSRKEY2 consists of user entered TCFLASH permission key bytes 8 to 11.

SCCFG_TCFPUSRKEY3 consists of user entered TCFLASH permission key bytes 12 to 15.

5.2.1.2 EEFLASH Permission User Key Register (SCCFG_EEFPUSRKEY0~3)

A 128-bit wide EEFLASH Permission User Key Register stores the user-entered permission key for the EEPROM Flash. A read access to this register returns zero.

EEFLASH Permission User Key Register (SCCFG_EEFPUSRKEY0~3)

Figure 5-3. EEFLASH Permission User Key Register (SCCFG_EEFPUSRKEY0)

SCCFG_EEFPUSRKEY0																															
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**
SCCFGEEFPUSRKEY0[31]	SCCFGEEFPUSRKEY0[30]	SCCFGEEFPUSRKEY0[29]	SCCFGEEFPUSRKEY0[28]	SCCFGEEFPUSRKEY0[27]	SCCFGEEFPUSRKEY0[26]	SCCFGEEFPUSRKEY0[25]	SCCFGEEFPUSRKEY0[24]	SCCFGEEFPUSRKEY0[23]	SCCFGEEFPUSRKEY0[22]	SCCFGEEFPUSRKEY0[21]	SCCFGEEFPUSRKEY0[20]	SCCFGEEFPUSRKEY0[19]	SCCFGEEFPUSRKEY0[18]	SCCFGEEFPUSRKEY0[17]	SCCFGEEFPUSRKEY0[16]	SCCFGEEFPUSRKEY0[15]	SCCFGEEFPUSRKEY0[14]	SCCFGEEFPUSRKEY0[13]	SCCFGEEFPUSRKEY0[12]	SCCFGEEFPUSRKEY0[11]	SCCFGEEFPUSRKEY0[10]	SCCFGEEFPUSRKEY0[9]	SCCFGEEFPUSRKEY0[8]	SCCFGEEFPUSRKEY0[7]	SCCFGEEFPUSRKEY0[6]	SCCFGEEFPUSRKEY0[5]	SCCFGEEFPUSRKEY0[4]	SCCFGEEFPUSRKEY0[3]	SCCFGEEFPUSRKEY0[2]	SCCFGEEFPUSRKEY0[1]	SCCFGEEFPUSRKEY0[0]
R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-4. EEFLASH Permission User Key Register (SCCFG_EEFPUSRKEY0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SCCFGEEFPUSRKEY0	EEFLASH User Permission Key If the [126:0] bits of user entered EEFLASH Permission Key (SCCFG_EEFPUSRKEY[126:0]) matches with the [126:0] bits of Nominal EEFLASH Permission Key (SCCFG_EEFPSYSKEY[126:0]) then SCCFG_EEFPUSRKEY[127] bit value determines the EEFLASH permission set. SCCFG_EEFPUSRKEY[127] is '0': Primary permission set is activated (default) '1': Secondary permission set is activated

Note:

Four 32-bit registers are present SCCFG_EEFPUSRKEY0, SCCFG_EEFPUSRKEY1, SCCFG_EEFPUSRKEY2, and SCCFG_EEFPUSRKEY3 to accommodate 128-bit user entered EEFLASH permission key.

SCCFG_EEFPUSRKEY0 consists of user entered EEFLASH permission key bytes 0 to 3.

SCCFG_EEFPUSRKEY1 consists of user entered EEFLASH permission key bytes 4 to 7.

SCCFG_EEFPUSRKEY2 consists of user entered EEFLASH permission key bytes 8 to 11.

SCCFG_EEFPUSRKEY3 consists of user entered EEFLASH permission key bytes 12 to 15.

5.2.2 System Interface registers

5.2.2.1 Security Control Register (SCCFG_CTRL)

SCCFG_CTRL register controls software enabling of JTAG access. SCCFG_CTRL register is writable only once, software can write into this register to enable JTAG access. A write access to the already written SCCFG_CTRL generates a bus error response. A read access returns register content.

Security Control Register (SCCFG_CTRL)

Figure 5-4. Security Control Register (SCCFG_CTRL)

SCCFG_CTRL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	JTAGSWEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-5. Security Control Register (SCCFG_CTRL) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	JTAGSWEN	Debug Software Enable This bit is used to enable JTAG access by the software. Software can write into this register only once to enable JTAG access, if SCCFG_STAT1:JTAGSWEN bit is set to '1'. The default value of SCCFG_CTRL:JTAGSWEN is '0'. '0': JTAG access is disabled by the software '1': JTAG access is enabled by the software

5.2.2.2 Security Status Register 0 (SCCFG_STAT0)

SCCFG_STAT0 register is used to indicate status of Flash permission key comparison and link key comparison. This is a read-only register. A read access to the SCCFG_STAT0 register returns the register content. A write access to this register generates a bus error response.

Security Status Register 0 (SCCFG_STAT0)

Figure 5-5. Security Status Register 0 (SCCFG_STAT0)

SCCFG_STAT0																															
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**
read0	read0	read0	read0	read0	read0	read0	EEFLNKOK	read0	read0	read0	read0	read0	read0	TCFLNKOK[1]	TCFLNKOK[0]	read0	read0	read0	read0	read0	EEFPSLOCK	EEFPS	EEFPEN	read0	read0	read0	read0	read0	TCFPSLOCK	TCFPS	TCFPEN
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-6. Security Status Register 0 (SCCFG_STAT0) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	EEFLNKOK	EEFLASH Link Key Comparison Status bit This bit indicates the status of EEFLASH link key comparison. '0': Read and execute accesses to EEFLASH are denied '1': Access to EEFLASH is allowed,
[23:18]	read0	-
[17:16]	TCFLNKOK	TCFLASH Link Keys Comparison Status bit These bits indicate the status of each TCFLASH macro link key comparison. TCFLNKOK[0] represents status of link key comparison for TCFLASH macro 0. '0': Read and execute accesses to TCFLASH macro 0 are denied '1': Access to TCFLASH macro 0 is allowed, TCFLNKOK[1] represents status of link key comparison for TCFLASH macro 1. '0': Read and execute accesses to TCFLASH macro 1 are denied '1': Access to TCFLASH macro 1 is allowed
[15:11]	read0	-

Table 5-6. Security Status Register 0 (SCCFG_STAT0) bits

Bit Position	Bit Field Name	Bit Description
[10]	EEFPSLOCK	EEFLASH Permission Key Lock bit This bit indicates whether EEFLASH permission key entry is locked or not. This bit is set to '1' if wrong key is written to SCCFG_EEFPUSRKEY register. '0': EEFLASH permission key entry is not locked. SCCFG_EEFPUSRKEY[127] bit selects active permission set '1': EEFLASH permission key entry is locked. Further key entry is ignored and current active permission set is retained until next hard reset
[9]	EEFPS	EEFLASH Permission Set bit This bit indicates which permission set is active for EEFLASH. This bit reflects value of SCCFG_EEFPUSRKEY[127] bit, if user entered EEFLASH permission key (SCCFG_EEFPUSRKEY) matches with Nominal EEFLASH Permission Key. '0': EEFLASH Permission Set zero is active '1': EEFLASH Permission Set one is active
[8]	EEFPEN	EEFLASH Permission Set Enable bit This bit indicates whether EEFLASH permission sets are enabled or not. This bit is set as per configuration done in SDR. If it is enabled, then switching between the EEFLASH permission sets is possible. '0': EEFLASH permission sets are disabled '1': EEFLASH permission sets are enabled
[7:3]	read0	-
[2]	TCFPSLOCK	TCFLASH Permission Key Lock bit This bit indicates whether TCFLASH permission key entry is locked or not. This bit is set to '1' if wrong key is written to SCCFG_TCFPUSRKEY register. '0': TCFLASH permission key entry is not locked. SCCFG_TCFPUSRKEY[127] bit selects active permission set '1': TCFLASH permission key entry is locked. Further key entry is ignored and current active permission set is retained until next hard reset
[1]	TCFPS	TCFLASH Permission Set bit This bit indicates which permission set is active for TCFLASH. This bit reflects value of SCCFG_TCFPUSRKEY[127] bit, if user entered TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY) matches with Nominal TCFLASH Permission Key. '0': TCFLASH Permission Set zero is active '1': TCFLASH Permission Set one is active

Table 5-6. Security Status Register 0 (SCCFG_STAT0) bits

Bit Position	Bit Field Name	Bit Description
[0]	TCFPEN	TCFLASH Permission Sets Enable bit This bit indicates whether TCFLASH permission sets are enabled or not. This bit is set as per configuration done in SDR. If it is enabled, then switching between the TCFLASH permission sets is possible. By default TCFLASH permission sets are disabled. '0': TCFLASH permission sets are disabled '1': TCFLASH permission sets are enabled

5.2.2.3 Security Status Register 1 (SCCFG_STAT1)

SCCFG_STAT1 register is used to indicate the status of device features (e.g. Flash parallel programming, serial communication etc.) and configuration lock/unlock. This is a read-only register. A read access to the SCCFG_STAT1 register returns the register content. A write access to this register generates a bus error response.

Security Status Register 1 (SCCFG_STAT1)

Figure 5-6. Security Status Register 1 (SCCFG_STAT1)

SCCFG_STAT1																															
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**
read0	read0	read0	read0	read0	read0	read0	CFGLOCK	read0	read0	read0	read0	read0	read0	read0	reserved	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	EEFCEEN	TCFCEEN	FPPEN	read0	JTAGSWEN	JTAGEN
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	-	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R0	R	R
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1

Table 5-7. Security Status Register 1 (SCCFG_STAT1) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	CFGLOCK	Configuration Register Lock bit This bit indicates whether write access to Security Checker Configuration Registers (SCCFG_CTRL, SCCFG_GPREG, SCCFG_SECKEY, SCCFG_TCFPUSRKEY, and SCCFG_EEFPUSRKEY) is locked or unlocked. '0': SCCFG_UNLCK register is written with unlock value. i.e. write access is allowed to above configuration registers '1': SCCFG_UNLCK register is written with lock value. i.e. write access is not allowed to above configuration registers
[23:17]	read0	-
[16]	reserved	-
[15:8]	read0	-
[7:6]	read0	-
[5]	EEFCEEN	EEFLASH Chip Erase Enable bit This bit indicates whether EEFLASH chip erase is enabled or not. This bit is set as per the configuration done in SDR. '0': EEFLASH chip erase is disabled '1': EEFLASH chip erase is enabled

Table 5-7. Security Status Register 1 (SCCFG_STAT1) bits

Bit Position	Bit Field Name	Bit Description
[4]	TCFCEEN	TCFLASH Chip Erase Enable bit This bit indicates whether TCFLASH chip erase is enabled or not. This bit is set as per the configuration done in SDR. '0': TCFLASH chip erase is disabled '1': TCFLASH chip erase is enabled
[3]	FPPEN	Flash Parallel Programming Enable bit This bit indicates whether Flash parallel programming mode can be entered in board test mode or not. This bit is set as per the configuration done in SDR. '0': Flash parallel programming mode can not be entered in board test mode '1': Flash parallel programming mode can be entered in board test mode
[2]	read0	-
[1]	JTAGSWEN	JTAG Software Enable bit This bit indicates whether software enabling of JTAG access is allowed or not. This bit is set as per the configuration done in SDR. By default software enabling of JTAG access is allowed. '0': Software enabling of JTAG access is not allowed (i.e. settting SCCFG_CTRL:JTAGSWEN bit to '1' is not allowed) '1': Software enabling of JTAG access is allowed (i.e. setting SCCFG_CTRL:JTAGSWEN bit to '1' is allowed)
[0]	JTAGEN	JTAG Enable bit This bit indicates whether JTAG access is enabled after BootROM execution or not. This bit is set as per the configuration done in SDR. By default JTAG access is enabled. '0': JTAG access is disabled '1': JTAG access is enabled

Note: For more details on availability of features of SCCFG_STAT1:EEFCEEN, SCCFG_STAT1:TCFCEEN, SCCFG_STAT1:FPPEN, SCCFG_STAT1:SCMEN bits refer to device specific datasheet.

5.2.2.4 Security Status Register 2 (SCCFG_STAT2)

SCCFG_STAT2 register is used to indicate the status of device security. This is a read- only register. A read access to the SCCFG_STAT2 register returns the register content. A write access to this register generates a bus error response.

Security Status Register 2 (SCCFG_STAT2)

Figure 5-7. Security Status Register 2 (SCCFG_STAT2)

SCCFG_STAT2																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGRDY	read0	read0	read0	read0	read0	SEC	SECLOCK	SECEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Table 5-8. Security Status Register 2 (SCCFG_STAT2) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	DBGRDY	Debug Ready bit This bit indicates whether the device is ready for debug access or not. '0': System is not ready for debug access (default) '1': System is ready for debug access
[7:3]	read0	-
[2]	SEC	Security bit This bit indicates whether security key can be entered or not for security enabled device. This bit switches to zero, after user enters security key, as security key can be entered only one time after a hard reset. This bit switches to zero, if manipulation is detected during BootROM execution or invalid key is entered. By default the bit is one, that means a key can be entered. '0': Security key cannot be entered '1': Security key can be entered

Table 5-8. Security Status Register 2 (SCCFG_STAT2) bits

Bit Position	Bit Field Name	Bit Description
[1]	SECLOCK	Device Security Lock bit. This bit indicates whether valid security key is entered or not. In other words, this bit indicates whether a secured device is currently locked or unlocked. If it is locked, then the SCCFG_STAT2:SEC bit indicates whether it can be unlocked. '0': Valid security key is entered by the user, device is unlocked, or device is not security enabled '1': Invalid security key is entered by the user, device is locked
[0]	SECEN	Security Enable bit This bit indicates whether device is security enabled or not. '0': Device is not security enabled '1': Device is security enabled If the device is unsecured, the SECEN bit is reset to '0' after evaluation of the SDRs.

5.2.3 Device Security registers

5.2.3.1 Security Key Register (SCCFG_SECKEY0~3)

A 128-bit wide Security Key Register stores the user entered security key. This register is writable only once. A write access to an already written SCCFG_SECKEY register generates a bus error response. A read access to this register returns zero.

Security Key Register (SCCFG_SECKEY0~3)

Figure 5-8. Security Key Register (SCCFG_SECKEY0)

SCCFG_SECKEY0																															
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**
SCCFGSECKEY0[31]	SCCFGSECKEY0[30]	SCCFGSECKEY0[29]	SCCFGSECKEY0[28]	SCCFGSECKEY0[27]	SCCFGSECKEY0[26]	SCCFGSECKEY0[25]	SCCFGSECKEY0[24]	SCCFGSECKEY0[23]	SCCFGSECKEY0[22]	SCCFGSECKEY0[21]	SCCFGSECKEY0[20]	SCCFGSECKEY0[19]	SCCFGSECKEY0[18]	SCCFGSECKEY0[17]	SCCFGSECKEY0[16]	SCCFGSECKEY0[15]	SCCFGSECKEY0[14]	SCCFGSECKEY0[13]	SCCFGSECKEY0[12]	SCCFGSECKEY0[11]	SCCFGSECKEY0[10]	SCCFGSECKEY0[9]	SCCFGSECKEY0[8]	SCCFGSECKEY0[7]	SCCFGSECKEY0[6]	SCCFGSECKEY0[5]	SCCFGSECKEY0[4]	SCCFGSECKEY0[3]	SCCFGSECKEY0[2]	SCCFGSECKEY0[1]	SCCFGSECKEY0[0]
R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-9. Security Key Register (SCCFG_SECKEY0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SCCFGSECKEY0	Actual Security Key Actual Device Security Key is stored in this register. Full access (access to debug, trace, test modes, and Flash memories) is given if SCCFG_SECKEY0~3 matches with the Nominal Device Security Key. Reduced access (access to debug, trace, and test modes without Flash memories) is given if this key matches with the Nominal FFA Key.

Note:

Four 32-bit registers are present SCCFG_SECKEY0, SCCFG_SECKEY1, SCCFG_SECKEY2, and SCCFG_SECKEY3 to accommodate 128-bit SCCFG_SECKEY.

SCCFG_SECKEY0 consists of user Device Security Key bytes 0 to 3.

SCCFG_SECKEY1 consists of user Device Security Key bytes 4 to 7.

SCCFG_SECKEY2 consists of user Device Security Key bytes 8 to 11.

SCCFG_SECKEY3 consists of user Device Security Key bytes 12 to 15.

5.2.4 General registers

5.2.4.1 Module ID Register (SCCFG_MODID)

SCCFG_MODID register stores the module ID of the Security Checker. The actual value of the Module ID is listed in device datasheet. A write access to this register generates a bus error response. A read access to this register returns the module ID.

Module ID Register (SCCFG_MODID)

Figure 5-9. Module ID Register (SCCFG_MODID)

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

Table 5-10. Module ID Register (SCCFG_MODID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MODID	MODID This register identifies the particular version of the hardware used in the device. This register helps in developing software to the hardware version implemented in the device. **Note:** Refer to the device specific datasheet for the module ID number details.

5.2.4.2 Security Checker Unlock Register (SCCFG_UNLCK)

The SCCFG_UNLCK register is used for locking or unlocking the write accesses of SCCFG_CTRL, SCCFG_GPREG, SCCFG_SECKEY, SCCFG_TCFPUSRKEY, and SCCFG_EEFPUSRKEY registers of Security Checker. Writing a specific 32-bit unlock value (0x5ECACCE5) to the SCCFG_UNLCK register allows the write access to above- mentioned registers and writing a specific 32-bit lock value (0xA135331A, different from unlock value) locks the access to these registers. A write access to the SCCFG_UNLCK register with wrong lock/unlock value leads to bus error response. A read access to this register always returns zero.

Security Checker Unlock Register (SCCFG_UNLCK)

Figure 5-10. Security Checker Unlock Register (SCCFG_UNLCK)

SCCFG_UNLCK																															
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**
SCUNLCK[31]	SCUNLCK[30]	SCUNLCK[29]	SCUNLCK[28]	SCUNLCK[27]	SCUNLCK[26]	SCUNLCK[25]	SCUNLCK[24]	SCUNLCK[23]	SCUNLCK[22]	SCUNLCK[21]	SCUNLCK[20]	SCUNLCK[19]	SCUNLCK[18]	SCUNLCK[17]	SCUNLCK[16]	SCUNLCK[15]	SCUNLCK[14]	SCUNLCK[13]	SCUNLCK[12]	SCUNLCK[11]	SCUNLCK[10]	SCUNLCK[9]	SCUNLCK[8]	SCUNLCK[7]	SCUNLCK[6]	SCUNLCK[5]	SCUNLCK[4]	SCUNLCK[3]	SCUNLCK[2]	SCUNLCK[1]	SCUNLCK[0]
R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W	R0W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-11. Security Checker Unlock Register (SCCFG_UNLCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SCUNLCK	Security Checker Unlock Register This register is used for locking or unlocking the write accesses of fol- lowing registers of the Security Checker:SCCFG_CTRL, SCCFG_GPREG, SCCFG_SECKEY, SCCFG_TCFPUSRKEY, and SCCFG_EEFPUSRKEY. By default write access to the configuration registers is blocked.

5.2.4.3 General Purpose Register (SCCFG_GPREG0~1)

SCCFG_GPREG is used to exchange data between the debugger (via JTAG) and the system even for a secured device. An application can exchange data with the debugger e.g. a software revision can be provided to the debugger even if the debugging is not allowed. Writing a 32-bit unlock value to the SCCFG_UNLCK register allows the write access to this register.

General Purpose Register (SCCFG_GPREG0~1)

Figure 5-11. General Purpose Register (SCCFG_GPREG0)

SCCFG_GPREG0																															
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**
SCCFGGPREG0[31]	SCCFGGPREG0[30]	SCCFGGPREG0[29]	SCCFGGPREG0[28]	SCCFGGPREG0[27]	SCCFGGPREG0[26]	SCCFGGPREG0[25]	SCCFGGPREG0[24]	SCCFGGPREG0[23]	SCCFGGPREG0[22]	SCCFGGPREG0[21]	SCCFGGPREG0[20]	SCCFGGPREG0[19]	SCCFGGPREG0[18]	SCCFGGPREG0[17]	SCCFGGPREG0[16]	SCCFGGPREG0[15]	SCCFGGPREG0[14]	SCCFGGPREG0[13]	SCCFGGPREG0[12]	SCCFGGPREG0[11]	SCCFGGPREG0[10]	SCCFGGPREG0[9]	SCCFGGPREG0[8]	SCCFGGPREG0[7]	SCCFGGPREG0[6]	SCCFGGPREG0[5]	SCCFGGPREG0[4]	SCCFGGPREG0[3]	SCCFGGPREG0[2]	SCCFGGPREG0[1]	SCCFGGPREG0[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-12. General Purpose Register (SCCFG_GPREG0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	**SCCFGGPREG0**	**General Purpose Register 0** **This register is used to exchange data between the debugger (via JTAG) and the system even for a secured device.**

Note:

Two General Purpose Registers SCCFG_GPREG0 and SCCFG_GPREG1, each of 32-bit width exists to exchange data.

5.3 Operation of Security Checker

The following section describes the operation of Security Checker

Concept of Security Description Record (SDR)

The complete security in the device is controlled by the entries in the Security Description Record which is stored in the TCFLASH. EEPROM Flash contains another SDR which stores EEPROM Flash related security settings. Both SDR entries can be changed by the user. The various settings from the Security Description Record can be read from the status registers of the Security Checker. For more details of which feature can be enabled or disabled using the SDR refer to 5.2 Security Checker registers. For more details of application view to program SDR refer to Notes to program SDR.

Operation of Security Checker

Introduction

Purpose of the FCR4 Cluster device security is the protection of the device against unauthorized access to program code or data. In addition, protection schemes are used to protect the accesses to the internal Flash memories. The device and Flash security can be enabled/disabled by security settings programmed in SDR.

The Security Checker helps in achieving system security in two parts:

- Device security
- Flash security

Device security

The device security prevents access to the overall device by blocking the JTAG debugging interface and also core-sight trace interfaces by two 128 bit pre-programmed security keys:

- Primary key also referred to as the device security key
- Secondary key also referred to as the Field Failure Analysis key (FFA key)

The device security key protects the complete access to the device via the JTAG. It protects the connection of any debugger or any other low level JTAG access. This key is used to gain access to debug, trace, test modes and Flash memories of the system.

The FFA key provides access to debug, trace, test modes of the device except the Flash memories. This feature distinguishes the device security key from the FFA key. In case the device security key is same as FFA key, then full access is granted. This includes the Flash memories and by this the FFA key feature can be disabled. These pre-programmed security keys are configured in the SDR of the TCFLASH.

User can enter 128-bit actual security key (SCCFG_SECKEY) via the JTAG interface at any point of time during the operation of the device. In other words, there is no restriction when the user wants to gain access to the device via JTAG. Access can be gained by this process even when an existing application or program is currently running in the device.

The key comparison is enabled only if the device is security enabled (SCCFG_STAT2:SECEN = '1') and JTAG access is enabled by SDR settings (SCCFG_STAT1:JTAGEN = '1') or JTAG access is enabled by the software (SCCFG_CTRL:JTAGSWEN = '1') and 128-bit actual security key (SCCFG_SECKEY) is completely written.

The Security Checker compares the actual security key (SCCFG_SECKEY) with a pre- programmed 128-bit device security key and FFA key stored in the TCFLASH. This pre- programmed 128 bit keys are referred to as the nominal keys or nominal key values. Alternatively, the Security Checker module also provides a method to enter the actual security key from the system side. This key can also be programmed by a program or application running in any of the program memory of the device. This key can reach the device by a communication interface of the device. For e.g., the key can be supplied via the CAN interface and captured by an API running in the device and later transferred to the Security Checker for comparison with the nominal keys. Once the key comparison is successful in the Security Checker, access is granted for further debug or trace through the JTAG and trace ports.

The device security keys have an additional protection with being one time writable. The Security Checker accepts the keys only once till further hard resets. This helps in ensuring multiple key entry based attacks are not attempted. Once the correct 128-bit key is entered to the Security Checker a second time write of the key generates an error response to the core or to the JTAG port, depends from where the key is entered. The error response to the core reflects as a data abort in the program execution. The error response reported to the JTAG port would only interrupt the debugger but not the program execution.

The status of key comparison can be checked by reading SCCFG_STAT2:SECLOCK register bit. If wrong key is entered, the Security Checker blocks debug access to the complete device until the next occurrence of a hard reset.

Note: For details on JTAG access through DAP (Debug Access Port) refer to Arm CoreSight Components Technical Reference Manual.

Flash security

Two permission sets are supported by each Flash macro, the Primary Permission Set (PPS) and the Secondary Permission Set (SPS) for e.g. to support software updates in a secure and easy way. Both permission sets are configured in the SDR. Based on the configuration done in SDR, the Flash permission set is enabled/disabled.

If the Flash permission set is enabled, the application can switch between these two permission sets by entering Flash permission keys. The Secondary Permission Set can be activated by writing the Flash permission key to a dedicated register in Security Checker (SCCFG_TCFPUSRKEY or SCCFG_EEFPUSRKEY). Switching back to the Primary Permission Set is possible by writing the Flash permission key again. Both permission sets have the same level of protection because both must be activated by the Flash permission key.

Security Checker controls the switch between the two permission sets of TCFLASH and EEFLASH by two pre-programmed 128-bit permission keys stored in the SDRs of TCFLASH and EEFLASH. These pre-programmed 128-bit keys are referred to as the nominal keys or nominal key values.

User can enter 128-bit Flash user permission key (SCCFG_TCFPUSRKEY or SCCFG_EEFPUSRKEY) via internal bus (e.g. the key can be supplied via the CAN interface) at any point of time during the operation of the device. In other words, there is no restriction when the user wants to switch between the Flash permission sets.

If Flash permission set is disabled in SDR configuration (i.e. SCCFG_STAT0:IFPEN = '0' for TCFLASH and SCCFG_STAT0:DFPEN = '0' for EEPROM Flash), then Flash full (read, write, and execute) access is granted and Flash permission key comparison is also disabled.

Once the Flash permission key comparison is enabled, the Flash security compares the 127 LSBs of TCFLASH user permission key (SCCFG_TCFPUSRKEY[126:0]) with 127 LSBs of pre- programmed TCFLASH nominal permission key.

When these bits match, SCCFG_TCFPUSRKEY[127] bit selects active permission set. When SCCFG_TCFPUSRKEY[127] bit is zero, Primary Permission Set is applied or when SCCFG_TCFPUSRKEY[127] bit is one, Secondary Permission Set is applied to TCFLASH.

The status of key comparison can be checked by reading SCCFG_STAT0:TCFPSLOCK register bit. Incase of wrong key entry, further key entry by user is ignored and the existing active permission set applies until next hard reset.

- The EEFLASH User Permission Key comparison is same as TCFLASH as explained above

Note:

For details about Flash access control refer to TCFLASH and EEPROM Flash chapters in hardware manual.

Apart from device security and Flash security, the Security Checker also supports following functions:

- Based on Flash link key comparison status (SCCFG_STAT0:TCFLNKOK[1:0], SCCFG_STAT0:EEFLNKOK), Security Checker either retains the active access permission to Flash memories or blocks read and execute access
- Security Checker provides the status, whether Flash parallel programming (SCCFG_STAT1:FPPEN), TCFLASH chip erase (SCCFG_STAT1:TCFCEEN), EEFLASH chip erase (SCCFG_STAT1:EEFCEEN) etc. are enabled or disabled to the system and Flash interface as per the configuration done in SDR. For more details on the availability of this functionality refer to the device-specific datasheet
- On all secured and unsecured devices two general purpose registers are provided in the Security Checker for exchange of data between internal program and the JTAG interface. For e.g. an application program can provide a software revision to the debugger using these registers even if device debug is blocked by the Security Checker

Table 5-13. Security key overview

Key	Size	Functional description
Device security key	128-bit	Gain access to debug, trace, test modes, and Flash memories of the system.
Field Failure Analysis key	128-bit	Gain access to debug, trace, test modes without Flash memory access of the system.
TCFLASH permission key	128-bit	Toggles access permissions for TCFLASH.
EEFLASH permission key	128-bit	Toggles access permissions for EEPROM Flash.

Debug security configuration by JTAG access, JTAG software access, and device security is given in the table below:

Table 5-14. Debug security configuration options

SCCFG_STAT1: JTAGEN	SCCFG_STAT1: JTAGSWEN	SCCFG_STAT2: SECEN	JTAG key entry enabled	Software can enable JTAG access	Key entry required for debug access
'0'	'0'	--	No	No	--
'0'	'1'	'0'	No	Yes	No
'0'	'1'	'1'	No	Yes	Yes
'1'	'0'	'0'	Yes	No	No
'1'	'0'	'1'	Yes	No	Yes
'1'	'1'	'0'	Yes	Yes	No
'1'	'1'	'1'	Yes	Yes	Yes

SCCFG_STAT1:JTAGEN, SCCFG_STAT1:JTAGSWEN, and SCCFG_STAT2:SECEN bits are set/ reset as per the configuration done in SDR.

The default configuration for an empty device is SCCFG_STAT1:JTAGEN = '1', SCCFG_STAT1:JTAGSWEN = '1', and SCCFG_STAT2:SECEN = '1'.

Note: For description about SCCFG_STAT1:JTAGEN, SCCFG_STAT1:JTAGSWEN, and SCCFG_STAT2:SECEN register bit refer to 5.2 Security Checker registers.

5.4 Notes on using Security Checker

The section explains the sequence of steps to be followed by the programmer to program SDR, to gain debug access, and to change Flash permission set.

Notes to program SDR

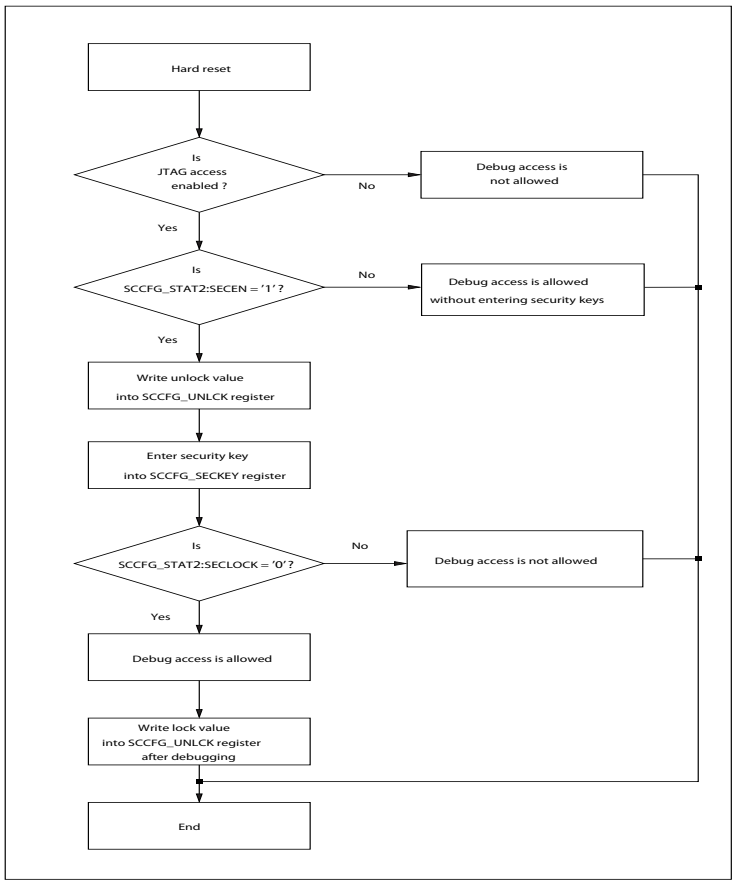
- Program or configure nominal device security key, FFA key into SDR
- Program nominal Flash permission system keys
- Program Flash link keys
- Program SDR to enable or disable other device security features
- Program primary and secondary access permission sets for both TCFLASH and EEPROM Flash memories
- Enable or disable device security and Flash permission sets

Notes to gain debug access

- If the JTAG debug access is disabled (SCCFG_STAT1:JTAGEN = '0'), then the software can enable the debug access by setting SCCFG_CTRL:JTAGSWEN bit to '1', if writing into SCCFG_CTRL is allowed in the SDR settings (i.e. SCCFG_STAT1:JTAGSWEN = '1')
- Before writing actual security key (SCCFG_SECKEY), the software must unlock configuration registers by writing unlock value (0x5ECACCE5) into SCCFG_UNLCK register
- 128-bit actual security key must always be written completely to enable key comparison and the key should always be written by 32-bit access only
- SCCFG_STAT2: SECLOCK bit indicates the status of key comparison
- Software must block write access to configuration registers by writing lock value (0xA135331A) into SCCFG_UNLCK register

Note: For details on JTAG access through DAP (Debug Access Port) refer to Arm CoreSight Components Technical Reference Manual.

Figure 5-12. Flow to gain debug access

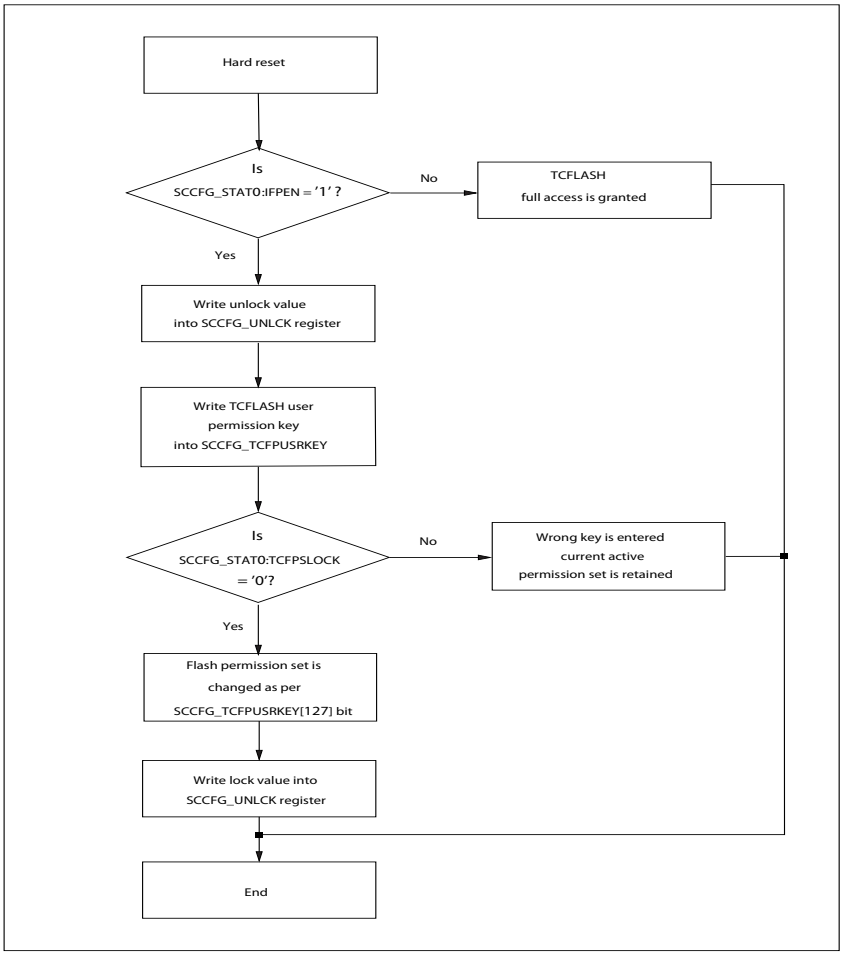


Notes on using TCFLASH permission key

- Software can check the SCCFG_STAT0:IFPEN bit to know whether TCFLASH permission set is enabled or not
- Before writing TCFLASH user permission key (SCCFG_TCFPUSRKEY), software must unlock configuration registers by writing unlock value into SCCFG_UNLCK register
- 128-bit TCFLASH user permission key should always be written completely to enable key comparison and the key should always be written by 32-bit access only
- SCCFG_STAT0:TCFPSLOCK bit indicates the status of key comparison
- Software must block write access to configuration registers by writing lock value into SCCFG_UNLCK register
- For 64-bit read out mode (which is always used in current device), in case of small sector access, two sectors (even and odd) are inter-leaved inside Flash macro and they share same access permissions. Due to common permissions used for interleaved sectors, permission bits of interleaved sectors are ignored (e.g. iIF_PERM_SET [5:3], access permission bits of SA1 (interleaved sector of SA0) are ignored).
- In case of big sector access, apart from interleaving inside Flashmacro, Flash-A and Flash-B are interleaved in addressing, thus 4 sectors are interleaved in total, sharing the same permissions. Also in case of big sectors, due to the reason that Flash-A shall always be accessible (as nominal link keys are stored in Flash-A), Flash-B permission is applied for Flash-A (thus big sector permission bits of Flash-A are ignored).

Notes on using EEFLASH permission key is same as TCFLASH.

Figure 5-13. Flowchart for TCFLASH permission key entry



6. Memory Protection Unit for AXI

This chapter explains the functions and operations of the Memory Protection Unit for AMBA AXI protocol bus (MPU AXI).

6.1 Outline of MPU AXI

This section describes the features and the block diagram of the MPU AXI.

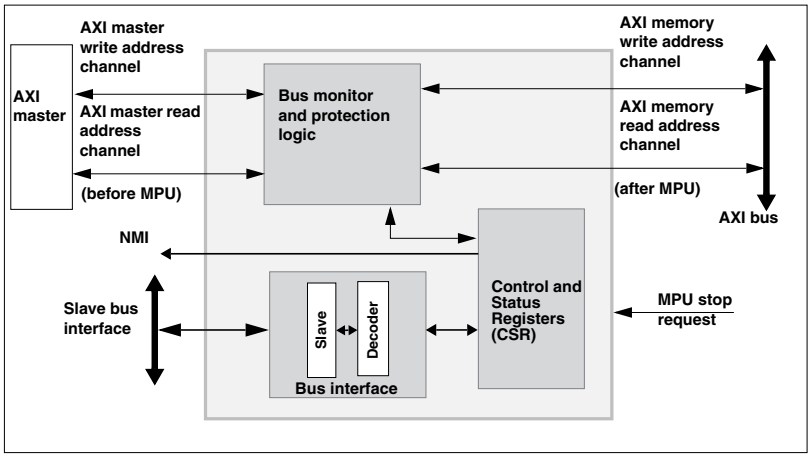
Features of the MPU AXI

The MPU AXI module monitors the accesses from AXI masters and checks each access against an authorized set of access permissions. Access permissions ('permission attributes' here onwards) are defined by 'access permissions' bits. These bits are explained in MPU access permissions. MPU AXI provides eight regions and one background region. Each region has corresponding access permission bits that defines the permission attributes for that particular region. Any unauthorized access to memory space is flagged using Non-Maskable Interrupt. MPU AXI also collects information about the unauthorized bus access and stores it in its internal registers.

- Each of the eight regions in MPU AXI is specified using corresponding start address and end address
- Background region covers entire 4 GB address space
- On unauthorized access MPU AXI generates an NMI to the CPU
- MPU AXI collects information about the AXI master bus access that caused unauthorized access
- Supports 8-bit, 16-bit, and 32-bit bus accesses for configuration of registers in MPU AXI
- Supports lock, unlock feature for protection of registers from illegal write accesses
- Modification of registers in MPU AXI is only allowed in privileged mode
- Supports MPU stop feature that allows blocking of all the accesses to memory space
- Optionally supports privileged mode overwrite feature that allows overwrite of privilege attribute of memory side AXI interface

Block diagram of MPU AXI

Figure 6-1. Block diagram of MPU AXI



Bus interface

The bus masters can access the MPU AXI module through its slave bus interface.

Bus monitor and protection logic

This logic monitors the transaction on AXI master interfaces. It finds out the region/s where the current transfer belongs to and then applies permissions based on the region match. It signals any permission violation to CSR logic that in turn generates NMI interrupt. All transactions on AXI master interfaces (including the transfer that caused permission violation) are blocked until NMI is cleared.

Control and Status Registers

The operation of MPU AXI can be controlled and monitored through its Control and Status Registers (CSR). For details of the CSR refer to 6.2 MPU AXI registers.

6.2 MPU AXI registers

The MPU AXI module contains various registers to configure its operation, to monitor its status and to read the information it has collected from the AXI master interfaces at the time of the memory protection violation.

The MPU AXI module is allocated 1 KB of MCU address space for mapping the Configuration and Status Registers (i.e. CSRs). The address area allocated to MPU AXI and the Control and Status Registers in MPU AXI are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of MPU AXI

The following registers are available for MPU AXI:

- MPU AXI Control Register (MPUXn_CTRL0)
- MPU AXI NMI Enable Register (MPUXn_NMIEN)
- MPU AXI Write Error Control Register (MPUXn_WERRC)
- MPU AXI Write Error Address Register (MPUXn_WERRA)
- MPU AXI Read Error Control Register (MPUXn_RERRC)
- MPU AXI Read Error Address Register (MPUXn_RERRA)
- MPU AXI Region Control Registers (MPUXn_CTRL1~8)
- MPU AXI Start Address Registers (MPUXn_SADDR1~8)
- MPU AXI End Address Registers (MPUXn_EADDR1~8)
- MPU AXI Unlock Register (MPUXn_UNLOCK)
- MPU AXI Module ID Register (MPUXn_MID)

Memory layout of MPU AXI registers

Table 6-1. Memory layout of MPU AXI registers

Offset	+3	+2	+1	+0
0x00000000	MPUXn_CTRL0 00000000 00000000 00000001 00000000			
0x00000004	MPUXn_NMIEN 00000000 00000000 00000000 00000001			
0x00000008	MPUXn_WERRC 00000000 00000000 00000XXX XXXXXXX0			
0x0000000C	MPUXn_WERRA XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000010	MPUXn_RERRC 00000000 00000000 00000XXX XXXXXXX0			
0x00000014	MPUXn_RERRA XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000018	MPUXn_CTRL1 00000000 00000000 00000000 00000000			
0x0000001C	MPUXn_SADDR1 00000000 00000000 00000000 00000000			
0x00000020	MPUXn_EADDR1 00000000 00000000 00000000 01111111			

Table 6-1. Memory layout of MPU AXI registers

Offset	+3	+2	+1	+0
0x00000024	MPUXn_CTRL2 00000000 00000000 00000000 00000000			
0x00000028	MPUXn_SADDR2 00000000 00000000 00000000 00000000			
0x0000002C	MPUXn_EADDR2 00000000 00000000 00000000 01111111			
0x00000030	MPUXn_CTRL3 00000000 00000000 00000000 00000000			
0x00000034	MPUXn_SADDR3 00000000 00000000 00000000 00000000			
0x00000038	MPUXn_EADDR3 00000000 00000000 00000000 01111111			
0x0000003C	MPUXn_CTRL4 00000000 00000000 00000000 00000000			
0x00000040	MPUXn_SADDR4 00000000 00000000 00000000 00000000			
0x00000044	MPUXn_EADDR4 00000000 00000000 00000000 01111111			
0x00000048	MPUXn_CTRL5 00000000 00000000 00000000 00000000			
0x0000004C	MPUXn_SADDR5 00000000 00000000 00000000 00000000			
0x00000050	MPUXn_EADDR5 00000000 00000000 00000000 01111111			
0x00000054	MPUXn_CTRL6 00000000 00000000 00000000 00000000			
0x00000058	MPUXn_SADDR6 00000000 00000000 00000000 00000000			
0x0000005C	MPUXn_EADDR6 00000000 00000000 00000000 01111111			
0x00000060	MPUXn_CTRL7 00000000 00000000 00000000 00000000			
0x00000064	MPUXn_SADDR7 00000000 00000000 00000000 00000000			
0x00000068	MPUXn_EADDR7 00000000 00000000 00000000 01111111			
0x0000006C	MPUXn_CTRL8 00000000 00000000 00000000 00000000			

Table 6-1. Memory layout of MPU AXI registers

Offset	+3	+2	+1	+0
0x00000070	MPUXn_SADDR8 00000000 00000000 00000000 00000000			
0x00000074	MPUXn_EADDR8 00000000 00000000 00000000 01111111			
0x00000078	MPUXn_UNLOCK 00000000 00000000 00000000 00000000			
0x0000007C	MPUXn_MID 00000000 00000000 00000000 00000000			

6.2.1 MPU AXI Control Register (MPUXn_CTRL0)

MPU AXI Control Register can be used by the software to configure the MPU AXI. This register provides a bit to enable the MPU AXI monitoring and protection function. It also provides permission attributes for background region. It also provides controls for enabling or disabling of privileged mode overwrite feature and MPU stop feature. Lastly it provides MPU AXI lock status and controls the status for Non-Maskable Interrupt.

MPU AXI Control Register (MPUXn_CTRL0)

Figure 6-2. MPU AXI Control Register (MPUXn_CTRL0)

MPUXn_CTRL0																															
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**
read0	read0	read0	read0	read0	AP[2]	AP[1]	AP[0]	read0	read0	read0	read0	read0	read0	MPUENC	MPUEN	read0	read0	read0	PROT	POEN	MPUSTOPEN	MPUSTOP	LST	read0	read0	read0	read0	read0	read0	NMICL	NMI
R0	R0	R0	R0	R0	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	RWP	R	R0	R0	R0	RWP	RWP	RWP	R	R	R0	R0	R0	R0	R0	R0	R0WP	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 6-2. MPU AXI Control Register (MPUXn_CTRL0) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26:24]	AP	Access Permissions for Background Region These bits are used to control access permissions for background region. For detailed description of these bits refer to Table 6-13.
[23:18]	read0	-
[17]	MPUENC	MPU AXI Enable Control '0': MPU AXI monitoring and protection function is disabled '1': MPU AXI monitoring and protection function is enabled Read returns status of MPUENC bit. The region enable status to be effective takes some delay with reference to write on MPUENC bit. The actual enable status can be read through MPUXn_CTRL0:MPUEN bit.
[16]	MPUEN	MPU AXI Enable Status '0': MPU AXI monitoring and protection function is disabled. All accesses on AXI master write or read address channel interfaces are passed on to AXI memory write or read address channel interfaces without any protection '1': MPU AXI monitoring and protection function is enabled

Table 6-2. MPU AXI Control Register (MPUXn_CTRL0) bits

Bit Position	Bit Field Name	Bit Description
[15:13]	read0	-
[12]	PROT	Privilege Attribute When privileged mode overwrite feature is available and also MPUXn_CTRL0:POEN = '1', privilege mode attribute on AXI memory interfaces are controlled by this bit. '0': Non-privilege mode '1': Privilege mode
[11]	POEN	Privileged Mode Overwrite Feature Enable '0': Privileged mode overwrite feature is disabled '1': Privileged mode overwrite feature is enabled **Note:** For availability of privileged mode overwrite feature, refer to device specific datasheet.
[10]	MPUSTOPEN	Enable for MPU STOP Feature '0': MPU stop feature is disabled '1': MPU stop feature is enabled This bit along with MPU stop input controls the STOP status of MPU AXI.
[9]	MPUSTOP	MPU Stop Status '0': MPU AXI is not stopped '1': MPU AXI is stopped (i.e.MPUXn_CTRL0:MPUSTOPEN = '1' and MPU stop input is asserted). All accesses on AXI master memory write or read address channel are converted to FIXED type burst with predefined address **Note:** For availability of MPU stop feature, refer to device specific datasheet.
[8]	LST	MPU Lock Status '0': MPU AXI is unlocked, registers in MPU AXI can be written '1': MPU AXI is locked, no registers (other than MPUXn_UNLOCK register) in MPU AXI can be written
[7:2]	read0	-
[1]	NMICL	NMI Interrupt Clear '0': No effect '1': Clears the NMI interrupt flag Read returns '0'.
[0]	NMI	NMI Interrupt Flag This bit indicates that the memory protection violation is detected for an AXI transaction.

Note: The register bits MPUXn_CTRL0:AP, MPUXn_CTRL0:MPUSTOPEN, MPUXn_CTRL0:POEN and MPUXn_CTRL0:PROT in MPUXn_CTRL0 register can only be written when MPU is disabled (MPUXn_CTRL0:MPUEN = '0'). This also implies that when MPUXn_CTRL0:MPUEN = '1':

- 32-bit or 16-bit access to this register is not allowed
- 8-bit access is required to disable the MPU
- 8-bit access is required to clear the NMI interrupt flag

6.2.2 MPU AXI NMI Enable Register (MPUXn_NMIEN)

MPU AXI NMI Enable Register can be used by software to reset NMI enable bit. Default value of NMI enable bit is '1'. This bit can be reset by software only once after reset operation.

MPU AXI NMI Enable Register (MPUXn_NMIEN)

Figure 6-3. MPU AXI NMI Enable Register (MPUXn_NMIEN)

MPUXn_NMIEN																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMIEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 6-3. MPU AXI NMI Enable Register (MPUXn_NMIEN) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	NMIEN	NMI Interrupt Enable This bit decides whether the NMI interrupt flag is routed to NMI interrupt signal or not. '0': NMI interrupt flag does not trigger NMI interrupt signal '1': NMI interrupt flag triggers NMI interrupt signal The value of this bit can be changed only once after reset.

6.2.3 MPU AXI Write Error Control Register (MPUXn_WERRC)

This is a read-only register that provides the software the control information of AXI transaction on AXI master write address channel interface for which memory protection violation was detected. Software can read this register to get privileged mode, burst length, burst size, and burst type information of AXI transaction.

MPU AXI Write Error Control Register (MPUXn_WERRC)

Figure 6-4. MPU AXI Write Error Control Register (MPUXn_WERRC)

MPUXn_WERRC																													
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
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AWSIZE[2]	AWSIZE[1]	AWSIZE[0]	AWBURST[1]	AWBURST[0]	AWLEN[3]	AWLEN[2]	AWLEN[1]	AWLEN[0]	AWPROTPRIV	AWMPV
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	0

Table 6-4. MPU AXI Write Error Control Register (MPUXn_WERRC) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10:8]	AWSIZE	AXI Transaction Burst Size This bit provides the status of AWSIZE[2:0] signals of AXI transaction for which memory protection violation is detected.
[7:6]	AWBURST	AXI Transaction Burst Type This bit provides the status of AWBURST[1:0] signals of AXI transaction for which memory protection violation is detected.
[5:2]	AWLEN	AXI Transaction Burst Length This bit provides the status of AWLEN[3:0] signals of AXI transaction for which memory protection violation is detected.
[1]	AWPROTPRIV	AXI Transaction Privileged Mode This bit provides the status of AWPROT[0] signal of AXI transaction for which memory protection violation is detected.
[0]	AWMPV	AXI Write Memory Protection Violation This bit indicates that memory protection violation is detected on AXI write address channel. Writing '1' to MPUXn_CTRL0:NMICL bit clears AWMPV bit.

6.2.4 MPU AXI Write Error Address Register (MPUXn_WERRA)

This is a read-only register that provides the software the write address of AXI transaction on AXI master write address channel interface for which memory protection violation was detected.

MPU AXI Write Error Address Register (MPUXn_WERRA)

Figure 6-5. MPU AXI Write Error Address Register (MPUXn_WERRA)

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

Table 6-5. MPU AXI Write Error Address Register (MPUXn_WERRC) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	AWADDR	AXI Write Address Write address of AXI transaction for which memory protection violation is detected.

6.2.5 MPU AXI Read Error Control Register (MPUXn_RERRC)

This is a read-only register that provides the software the control information of AXI transaction on AXI master read address channel interface for which memory protection violation was detected. Software can read this register to get privileged mode, burst length, burst size, and burst type information of AXI transaction.

MPU AXI Read Error Control Register (MPUXn_RERRC)

Figure 6-6. MPU AXI Read Error Control Register (MPUXn_RERRC)

MPUXn_RERRC																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ARSIZE[2]	ARSIZE[1]	ARSIZE[0]	ARBURST[1]	ARBURST[0]	ARLEN[3]	ARLEN[2]	ARLEN[1]	ARLEN[0]	ARPROTPRIV	ARMPV
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	0

Table 6-6. MPU AXI Read Error Control Register (MPUXn_RERRC) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10:8]	ARSIZE	AXI Transaction Burst Size This bit provides the status of ARSIZE[2:0] signals of AXI transaction for which memory protection violation is detected.
[7:6]	ARBURST	AXI Transaction Burst Type This bit provides the status of ARBURST[1:0] signals of AXI transaction for which memory protection violation is detected.
[5:2]	ARLEN	AXI Transaction Burst Length This bit provides the status of ARLEN[3:0] signals of AXI transaction for which memory protection violation is detected.
[1]	ARPROTPRIV	AXI Transaction Privileged Mode This bit provides the status of ARPROT[0] signal of AXI transaction for which memory protection violation is detected.
[0]	ARMPV	AXI Read Memory Protection Violation This bit indicates that memory protection violation is detected on AXI read address channel. Writing '1' to MPUXn_CTRL0:NMICL bit clears ARMPV bit.

6.2.6 MPU AXI Read Error Address Register (MPUXn_RERRA)

This is a read-only register that provides the software the read address of AXI transaction on AXI master read address channel interface for which memory protection violation was detected.

MPU AXI Read Error Address Register (MPUXn_RERRA)

Figure 6-7. MPU AXI Read Error Address (MPUXn_RERRA)

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

Table 6-7. MPU AXI Read Error Address (MPUXn_RERRA) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ARADDR	AXI Read Address Read address of AXI transaction for which memory protection violation is detected.

6.2.7 MPU AXI Region Control Registers (MPUXn_CTRL1~8)

MPU AXI Region Control Register is used to specify access permission for a particular region. Software can also use this register for enabling or disabling the particular region. MPUXn_CTRL1 Control Register for Region 1 is explained below.

MPU AXI Region Control Register for Region 1 (MPUXn_CTRL1)

Figure 6-8. MPU AXI Region Control Register for region 1 (MPUXn_CTRL1)

MPUXn_CTRL1																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AP[2]	AP[1]	AP[0]	read0	read0	read0	read0	read0	read0	MPUENC	MPUEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	RWP	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-8. MPU AXI Region Control Register for region 1 (MPUXn_CTRL1) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10:8]	AP	Access Permissions These bits are used to control access permissions for region 1. **Note:** For detailed description of these bits refer to Table 6-13.
[7:2]	read0	-
[1]	MPUENC	Enable Control bit '0': Memory protection for region 1 is disabled '1': Memory protection for region 1 is enabled Read returns status of MPUENC bit. The region enable status to be effective takes some delay with reference to write on MPUENC bit. The actual enable status can be read through MPUXn_CTRL1:MPUEN bit.
[0]	MPUEN	Enable Status '0': Memory protection for region 1 is disabled '1': Memory protection for region 1 is enabled This is a read-only bit, writing to this bit does not have any effect.

Note:

Access permission bits (i.e. MPUXn_CTRL1~8:AP) can only be written when corresponding region is disabled (MPUXn_CTRL1~8:MPUEN = '0') or MPU is disabled (MPUXn_CTRL0:MPUEN = '0').

This also implies that when MPUXn_CTRL0:MPUEN = '1' and MPUXn_CTRL1~8:MPUEN = '1':

- 32-bit or 16-bit access to this register is not allowed
- 8-bit access is required to disable the MPU region

For the number of regions available in your device refer to the device datasheet.

6.2.8 MPU AXI Start Address Registers (MPUXn_SADDR1~8)

Each region in MPU AXI can be set by specifying start address and end address for that particular region. MPUXn_SADDR1~8 registers are used to specify start address for eight regions. Start address indicates the first address for that region. MPUXn_SADDR1 Start Address Register for Region 1 is explained below.

MPU AXI Start Address Register for Region 1 (MPUXn_SADDR1)

Figure 6-9. MPU AXI Start Address Register for Region 1 (MPUXn_SADDR1)

MPUXn_SADDR1																															
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**
SADDR[31]	SADDR[30]	SADDR[29]	SADDR[28]	SADDR[27]	SADDR[26]	SADDR[25]	SADDR[24]	SADDR[23]	SADDR[22]	SADDR[21]	SADDR[20]	SADDR[19]	SADDR[18]	SADDR[17]	SADDR[16]	SADDR[15]	SADDR[14]	SADDR[13]	SADDR[12]	SADDR[11]	SADDR[10]	SADDR[9]	SADDR[8]	SADDR[7]	SADDR[6]	SADDR[5]	SADDR[4]	SADDR[3]	SADDR[2]	SADDR[1]	SADDR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-9. MPU AXI Start Address Register for region 1 (MPUXn_SADDR1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SADDR	Start Address Start address for region 1

Note: MPUXn_SADDR1~8 registers can only be written when corresponding region is disabled (MPUXn_CTRL1~8:MPUEN = '0') or MPU is disabled (MPUXn_CTRL0:MPUEN = '0').

For the number of regions available in your device refer to the device datasheet.

6.2.9 MPU AXI End Address Registers (MPUXn_EADDR1~8)

Each region in MPU AXI can be set by specifying start address and end address for that particular region. MPUXn_EADDR1~8 registers are used to specify end addresses for eight regions. End Address indicates the last address for that region. MPUXn_EADDR1 End Address Register for Region 1 is explained below.

MPU AXI End Address Register for Region 1 (MPUXn_EADDR1)

Figure 6-10. MPU AXI End Address Register for Region 1 (MPUXn_EADDR1)

MPUXn_EADDR1																															
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**
EADDR[31]	EADDR[30]	EADDR[29]	EADDR[28]	EADDR[27]	EADDR[26]	EADDR[25]	EADDR[24]	EADDR[23]	EADDR[22]	EADDR[21]	EADDR[20]	EADDR[19]	EADDR[18]	EADDR[17]	EADDR[16]	EADDR[15]	EADDR[14]	EADDR[13]	EADDR[12]	EADDR[11]	EADDR[10]	EADDR[9]	EADDR[8]	EADDR[7]	EADDR[6]	EADDR[5]	EADDR[4]	EADDR[3]	EADDR[2]	EADDR[1]	EADDR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 6-10. MPU AXI End Address Register for Region 1 (MPUXn_EADDR1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EADDR	End Address End address for region 1

Note: MPUXn_EADDR1~8 registers can only be written when corresponding region is disabled (MPUXn_CTRL1~8:MPUEN = '0') or MPU is disabled (MPUXn_CTRL0:MPUEN = '0').

Refer to the device datasheet for the number of regions available in your device.

6.2.10 MPU AXI Unlock Register (MPUXn_UNLOCK)

The software can use this register to lock or unlock the MPU AXI registers for write access.

MPU AXI Unlock Register (MPUXn_UNLOCK)

Figure 6-11. MPU AXI Unlock Register (MPUXn_UNLOCK)

MPUXn_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-11. MPU AXI Unlock Register (MPUXn_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	MPU AXI Unlock The MPU AXI Unlock Register protects the MPU AXI module from being modified accidently by software. The MPU AXI registers cannot be written until this register has been written with a specific unlock value. The correct value for unlocking can be written only in privileged mode. Reading this register always returns a zero. To lock the MPU AXI again software must write another value specific to lock. Write access to MPU AXI registers without unlocking or writing a value other than the lock or unlock values to this register causes a protection error. **Notes:** 1. This register can not be written by 8-bit or 16-bit write access; any such access causes protection error. 2. For more details on lock and unlock values, refer to the device specific datasheet.

6.2.11 MPU AXI Module ID Register (MPUXn_MID)

This is a read-only register with a unique module identification number which identifies the version of the MPU AXI module used in the MCU.

MPU AXI Module ID (MPUXn_MID)

Figure 6-12. MPU AXI Module ID Register (MPUXn_MID)

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

Table 6-12. MPU AXI Module ID Register (MPUXn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID The MPU AXI module implemented in the device may vary from device to device. This register identifies the particular version of the hardware used in the device. This register helps in developing software to hardware version implemented in the device **Note:** For more details on module ID number, refer to device specific datasheet.

Note: Refer to the device specific datasheet for the module identification number of the MPU AXI module in the device.

6.3 Operation of MPU AXI

This section describes the operation of MPU AHB.

MPU AXI provides start address and end address for each of the eight regions. MPU AXI regions are defined with granularity of 128 bytes.

Start address specifies the first address of the region and is specified by registers MPUXn_SADDR1 to MPUXn_SADDR8 for region 1 to region 8 respectively. Since the region granularity is 128 bytes least significant 7 bits of start addresses will read 0.

End addresses are specified by registers MPUXn_EADDR1 to MPUXn_EADDR8 for region 1 to region 8 respectively. Least significant 7 bits of End Address Registers are read-only bits and will always read '1'. This ensures the granularity of 128 bytes.

AXI burst monitoring

Bus monitor and protection logic monitors AXI master 'write address channel' signals and 'read address channel' signals ('AXI master interfaces' here onwards). The transactions on AXI master interfaces are manipulated (if required) before they are passed on to AXI memory 'write address channel' signals and 'read address channel' signals ('AXI memory interfaces' here onwards).

AXI protocol supports following features

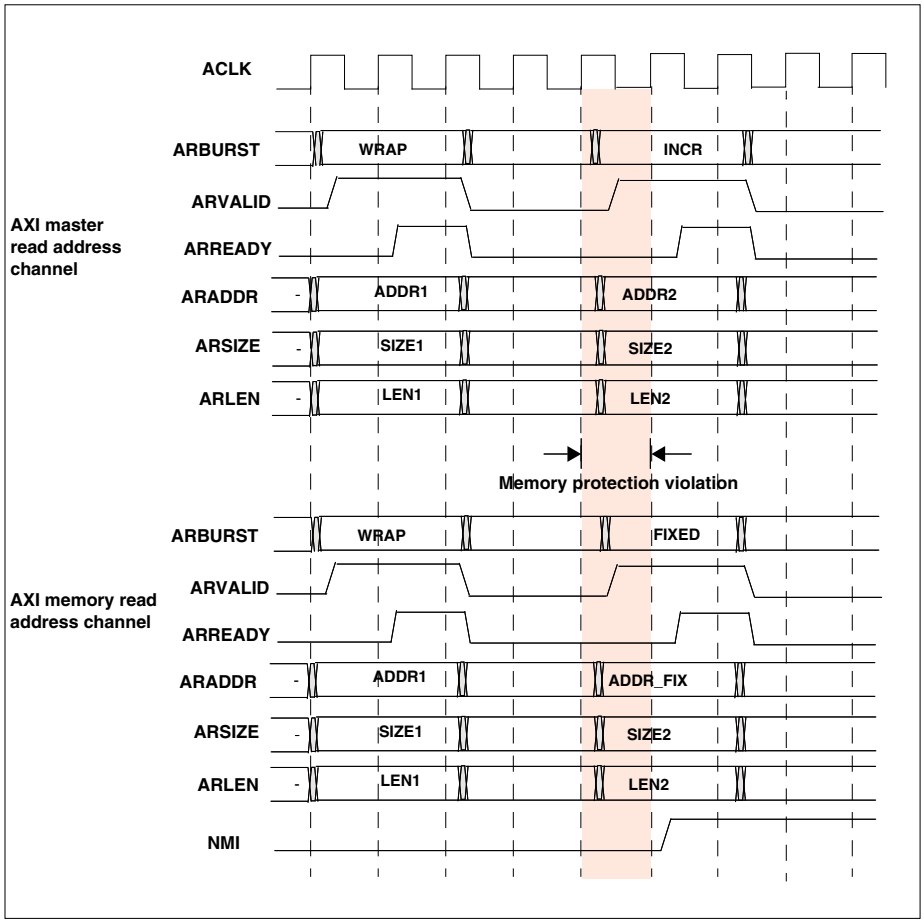
- Separate address or control and data phases
- Burst based transaction where only start address is issued
- Separate write and read address/control channels
- Separate write and read data channels

AXI master begins each burst by driving transaction control information and address of the first byte in the transaction. As the burst transaction progresses, AXI slave calculates the addresses of subsequent transfers in the burst. The AWLEN (write address channel signal), ARLEN (read address channel signal) specifies the number of data transfers for a burst transaction. Each burst can be of 1 to 16 transfers long. The AWSIZE, ARSIZE signals specifies the maximum number of data transfers in terms of bytes in each data transfer within a burst. The AXI protocol defines FIXED, incrementing and wrapping burst types.

As the addresses are defined in separate channels than data channels MPU AXI monitors and controls only address channels of AXI. If memory protection violation is detected entire burst transaction is manipulated to FIXED address burst with address of predefined value.

Figure 6-13 shows timings for read address channel signals. First burst is of WRAP type for which no memory protection violation was detected. Second burst is of INCR type for which memory protection violation was detected and hence MPU AXI manipulates the ARBURST to FIXED type and ARADDR to predefined value (shown as ADDR_FIX).

Figure 6-13. MPU AXI example timings



MPU AXI uses burst start address (AWADDR, ARADDR signals), burst type (AWBURST, ARBURST signals), burst length (AWLEN, ARLEN signals), and burst size (AWSIZE, ARSIZE signals) from AXI burst transaction to calculate the lowest address and the highest address of the burst and then both the addresses are used to find the region match.

For FIXED type of burst:

Lowest address = Highest address = Burst start address

For INCR type of burst:

Lowest address = Burst start address

Highest address = Burst start address + (num_bytes x num_transfers) – 1

where, num_bytes is number of bytes in each transfer of burst and num_transfers is number of transfers in the burst

For WRAP type of burst:

Lowest address = Wrap boundary

Highest address = Wrap boundary + (num_bytes x num_transfers) – 1

MPU access permissions

Region control registers in MPU AXI, MPUXn_CTRL1 to MPUXn_CTRL8 are used to control the access permission for region 1 to region 8 respectively. Also MPUXn_CTRL0 is used to control the access permission for background region.

Table 6-13 describes access permission bits (AP) in these registers and corresponding access permissions.

Table 6-13. Access permissions

AP bits	Access in privileged mode	Access in non- privileged mode	Comment
'000' (default)	No access	No access	All bus accesses are blocked and hence would generate memory protection violation.
'001'	read, write	No access	Reads and writes are permitted in privileged mode only. Any access in non-privileged mode would generate memory protection violation.
'010'	read, write	read only	Reads and writes are permitted in privileged mode. Only reads are permitted in non-privileged mode. Writes in non-privileged mode would generate memory protection violation.
'011'	read, write	read, write	All bus transfers are permitted. No memory protection violation is generated in this mode.
'100'	No access	No access	All bus accesses are blocked and hence generate memory protection violation.
'101'	read only	No access	Reads are permitted in privileged mode only. Writes in privileged mode and any access in non-privileged mode would generate memory protection violation.
'110'	read only	read only	Reads are permitted in privileged as well as non-privileged mode. Any write access would generate memory protection violation.
'111'	read, write	read, write	All bus transfers are permitted. No memory protection violation is generated in this mode.

MPU priority decision

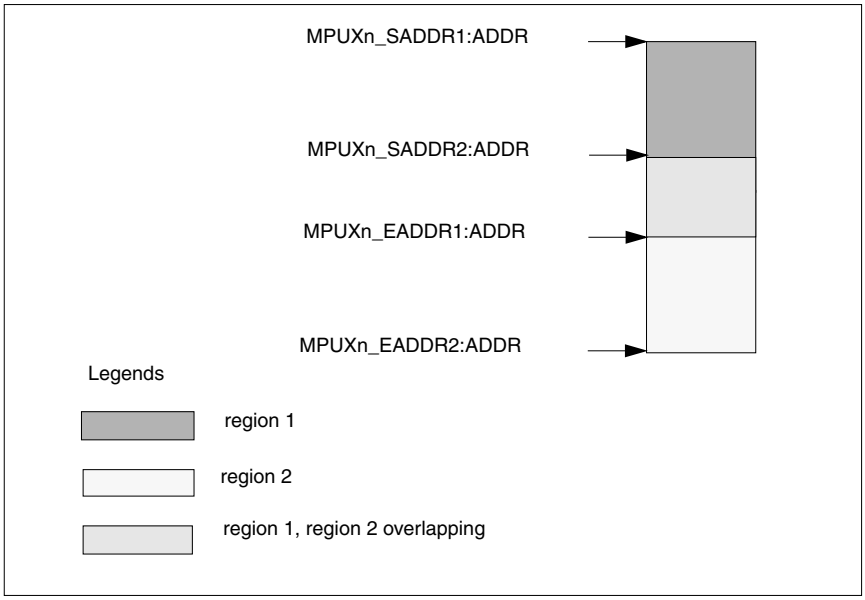
MPU AXI provides start address and end address for each region. Start address specifies the first address of the region and end address specifies the last address of the region. It is allowed in MPU AXI that the region may overlap each other. Hence it is also possible that an AXI transaction address may match with multiple regions.

For AXI protocol transaction following scenarios are possible.

1. Lowest address and highest address might fall in the same region.
2. Lowest address might match in one region, but highest address might match in some other region.

3. Scenario mentioned in point 1 here, happens for multiple regions due to region overlapping.
4. Scenario mentioned in point 2 here, happens for multiple regions due to region overlapping.

Figure 6-14. Example region overlapping



MPU AXI finds region match signals based on lowest address of AXI burst transaction and also separate region match signals based on highest address. Due to overlapping of regions it is possible lowest address and/or highest address may match with multiple regions.

MPU AXI supports upto eight regions. Each region is identified as region 1, region 2, region 3, and so on upto region 8. Region 8 is given highest priority, region 7 has second highest priority, and so on with region 1 as the second lowest priority region. Background region is the lowest priority region.

Whenever the AXI transaction lowest address matches with multiple regions the permissions corresponding to the highest priority (among all matching region) region is applied. Similarly, if AXI transaction highest address matches with multiple regions the permissions corresponding to highest priority (among all matching region) region is applied.

It is possible that lowest address and highest address may match with different highest priority regions. In this case the more restrictive permission is applied. Table 6-13 shows access permissions with corresponding restriction index for privileged mode access. Restriction index 1 has the most restrictive access and access permissions with restriction index 3 has the least restrictive access. This table is applied when privileged mode attribute of current transaction is of type privileged mode.

Similarly, Table 6-13 shows access permissions with corresponding restriction index for non- privileged mode access. This table is applied when privileged mode attribute of current transaction is of type non-privileged mode.

Table 6-14. Restrictive access permission matrix for privileged mode access

AP bits	Access in privileged mode	Restriction index
'000', '100'	No access	1
'101', '110'	read only	2
'001', '010', '011', '111'	read, write	3

Table 6-15. Restrictive access permission matrix for non-privileged mode access

AP bits	Access in non-privileged mode	Restriction index
'000', '001', '100', '101'	No access	1
'010', '110'	read only	2
'011', '111'	read, write	3

Bus monitor and protection logic

All transactions on the AXI master interfaces are monitored and checked for permitted access.

- Bus monitor and protection logic within MPU AXI compares the lowest address and highest address of current transaction with start addresses and end addresses of each region to find region match where the current transaction matches to among the eight defined regions
- As explained in MPU priority decision the AXI transaction address may match with multiple regions. The permission attributes of highest priority region are checked against the attributes of currently applied transaction from AXI master
- If the attributes of currently applied transaction are within permitted attributes, current transaction is passed on to the AXI memory interfaces
- If the attributes are not within permitted attributes current transaction is blocked. The Non Maskable Interrupt (MPUXn_CTRL0:NMI) flag is set. If the memory protection violation is detected on write address channel, the address and control information is stored in MPUXn_WERRA and MPUXn_WERRC registers respectively. If the memory protection violation is detected on read address channel, the address and control information is stored in MPUXn_RERRA and MPUXn_RERRC registers respectively. It is possible that memory protection violation is detected simultaneously on both the channels
- All further transaction are blocked until the MPUXn_CTRL0:NMI flag is cleared by software. Further monitoring of AXI master interfaces is also stalled until the MPUXn_CTRL0:NMI flag is cleared

When a transfer is blocked, MPU AXI does the following actions:

- Current transaction on AXI master is manipulated to FIXED address burst
- The transaction address is changed to predefined address. Check device specific datasheet for details of predefined address

MPU stop feature

Optionally MPU AXI supports MPU stop feature.

When this mode is enabled all accesses to memory space are blocked and MPU AXI does the following actions:

- Burst type signal is driven to FIXED type burst
- Burst address is driven to predefined FIXED address

For more details refer to MPU stop related register bits in 6.2.1 MPU AXI Control Register (MPUXn_CTRL0).

Privileged mode overwrite feature

Optionally MPU AXI supports privileged mode overwrite feature.

When this mode is enabled privileged mode attribute on the AXI memory interfaces are set by register bit setting MPUXn_CTRL0:PROT as explained in 6.2.1 MPU AXI Control Register (MPUXn_CTRL0).

Note: Bus monitor and protection logic for detecting memory protection violation uses privilege mode attribute on AXI master interfaces and not MPUXn_CTRL0:PROT bit setting even when privileged mode overwrite feature is enabled.

6.4 Notes on using MPU AXI

This section is the 'programmer's guide', which lists the usage notes for programming the MPU AXI module. It is recommended to read these guidelines before programming the MPU AXI module.

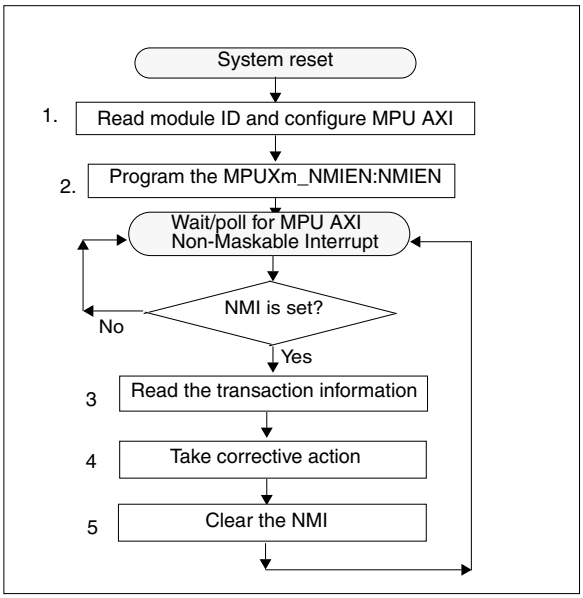
General usage notes

- Reserved bits return undefined values. The software programs shall be independent of the values read from the reserved register bits
- The MPU AXI supports storage of information of first memory protection violation on the bus. Therefore, once an NMI interrupt flag is set, the further monitoring of AXI master interface is stalled until MPUXn_CTRL0:NMI bit is cleared. This implies that any protection violation occurring on the bus while the MPUXn_CTRL0:NMI is set are simply ignored by MPU AXI. Software developers must therefore try to keep the ISR size small, so that the NMI interrupt of MPU AXI does not remain unattended for long

Steps in programming the MPU AXI module

Figure 6-15 gives the general steps a programmer shall follow while using the MPU AXI module.

Figure 6-15. Programmer's flowchart



1. After the system reset, the software detects the module ID number of MPU AXI, by reading the MPUXn_MID register. This helps it in identifying the attributes and capabilities supported by the MPU AXI module. Then the software configures MPU AXI by setting appropriate registers.
2. By default, MPU AXI propagates the MPUXn_CTRL0:NMI flag to the CPU through the Interrupt Controller. If polling mode is desired, the software can reset the MPUXn_NMIEN:NMIEN bit to '0'.

Note:

The MPUXn_NMIEN:NMIEN can be written only once after reset. Subsequent write accesses to this bit have no visible impact on the state of this bit.

3. When the NMI is triggered or is in polling mode, if the software detects during its polling cycle that the MPUXn_CTRL0:NMI status flag is set, the CPU is invoked. This would read the status information collected and stored by MPU AXI in its CSR.
4. The software diagnoses the information about the transaction for which memory protection violation was detected and initiates a corrective action (if any).

5. Once the software has processed the information from the status registers, it shall clear the MPUXn_CTRL0:NMI flag by writing a '1' to the MPUXn_CTRL0:NMICL bit. Clearing MPUXn_CTRL0:NMI flag ensures that the MPU AXI starts monitoring the AXI master interfaces again for checking memory protection violation.

Note: Software may clear the NMI flag before taking corrective action, hence steps 4 and 5 can be interchanged.

7. Memory Protection Unit for AHB

This chapter explains the function and operation of the Memory Protection Unit for the AMBA Advanced High Speed Bus (MPU AHB) in the Next Generation Microcontrollers.

7.1 Outline of the MPU AHB

This section describes the features and the block diagram of the MPU AHB.

Features of MPU AHB

The MPU AHB module monitors the accesses from AHB Masters and checks each access against an authorized set of Access Permissions. Access Permissions (known as “permission attributes” from here onwards) are defined by Access Permissions (AP) bits. These AP bits are explained in MPU access permissions.

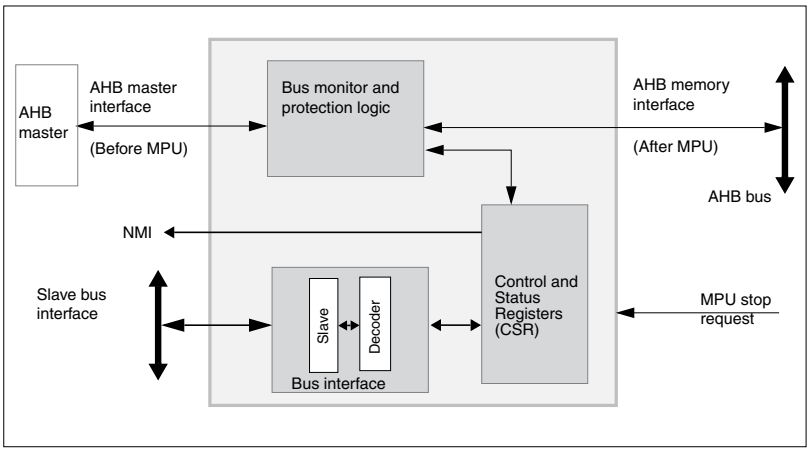
The MPU AHB provides eight regions and one background region. Each region has corresponding Access Permission bits that define the permission attributes for that particular region. Any unauthorized access to that memory space is flagged using a non-maskable interrupt. MPU AHB also collects information about unauthorized bus access and stores it in its internal registers.

The features of the MPU AHB module are listed in this section:

- Each of the eight regions in MPU AHB is specified using corresponding start address and end address
- The background region covers the entire 4GB address space
- With an unauthorized access, the MPU AHB generates an NMI to the CPU
- The MPU AHB collects information about the AHB Master bus access that caused the unauthorized access
- It supports 8-, 16- and 32-bit bus accesses for the configuration of the MPU AHB registers
- It supports lock and unlock feature to protect registers from illegal write accesses
- The modification of registers in the MPU AHB is only allowed in privileged mode
- MPU AHB registers can be written only after execution of the unlock sequence
- It supports an MPU STOP feature that allows blocking of all the accesses to memory space
- It optionally supports a privileged mode overwrite feature that allows the privilege attribute of the memory side AHB interface to be overwritten

Block diagram of the MPU AHB

Figure 7-1. Block diagram of MPU AHB



- Bus interface

  The bus masters can access the MPU AHB module through its slave bus interface.
- Bus monitor and protection logic

This logic monitors the transfers on AHB master interface bus. It finds out the region/s where the current transfer belongs to and then applies permissions based on the region match. It signals any permission violation using an NMI interrupt. All transfers on the AHB Master Interface (including the one that caused permission violation) are blocked until the NMI is cleared.

- Control and Status Registers

The operation of the MPU AHB can be controlled and monitored through its Control and Status Registers (CSR). For more details about the CSR refer to 7.2 MPU AHB registers

7.2 MPU AHB registers

The MPU AHB module contains various registers to configure its operation, to monitor its status and to read the information it has collected from the AHB master interface at the time of a memory protection violation.

The MPU AHB module is allocated 1 KB of MCU address space for mapping the Configuration and Status Registers (CSRs). The address area allocated to the MPU AHB and the CSRs in the MPU AHB are explained in this section.

The suffix ‘n’ in the register name indicates that the register is an instance ‘n’ of the module.

Registers of MPU AHB

The following registers are available for the MPU AHB:

- MPU AHB Control Register (MPUHn_CTRL0)
- MPU AHB NMI Enable Register (MPUHn_NMIEN)
- MPU AHB Memory Error Control Register (MPUHn_MERRC)
- MPU AHB Memory Error Address Register (MPUHn_MERRA)
- MPU AHB Region Control Registers (MPUHn_CTRL1~8)
- MPU AHB Start Address Registers (MPUHn_SADDR1~8)
- MPU AHB End Address Registers (MPUHn_EADDR1~8)
- MPU AHB Unlock Register (MPUHn_UNLOCK)
- MPU AHB Module ID Register (MPUHn_MID)

Memory layout of MPU AHB registers

Table 7-1. Memory layout of MPU AHB registers

Offset	+3	+2	+1	+0
0x00000000	MPUHn_CTRL0 00000000 00000000 00000001 00000000			
0x00000004	MPUHn_NMIEN 00000000 00000000 00000000 00000001			
0x00000008	MPUHn_MERRC 00000000 00000000 00000000 00000000			
0x0000000C	MPUHn_MERRA 00000000 00000000 00000000 00000000			
0x00000010	MPUHn_CTRL1 00000000 00000000 00000000 00000000			
0x00000014	MPUHn_SADDR1 00000000 00000000 00000000 00000000			
0x00000018	MPUHn_EADDR1 00000000 00000000 00000000 00011111			
0x0000001C	MPUHn_CTRL2 00000000 00000000 00000000 00000000			
0x00000020	MPUHn_SADDR2 00000000 00000000 00000000 00000000			
0x00000024	MPUHn_EADDR2 00000000 00000000 00000000 00011111			

Table 7-1. Memory layout of MPU AHB registers

Offset	+3	+2	+1	+0
0x00000028	MPUHn_CTRL3 00000000 00000000 00000000 00000000			
0x0000002C	MPUHn_SADDR3 00000000 00000000 00000000 00000000			
0x00000030	MPUHn_EADDR3 00000000 00000000 00000000 00011111			
0x00000034	MPUHn_CTRL4 00000000 00000000 00000000 00000000			
0x00000038	MPUHn_SADDR4 00000000 00000000 00000000 00000000			
0x0000003C	MPUHn_EADDR4 00000000 00000000 00000000 00011111			
0x00000040	MPUHn_CTRL5 00000000 00000000 00000000 00000000			
0x00000044	MPUHn_SADDR5 00000000 00000000 00000000 00000000			
0x00000048	MPUHn_EADDR5 00000000 00000000 00000000 00011111			
0x0000004C	MPUHn_CTRL6 00000000 00000000 00000000 00000000			
0x00000050	MPUHn_SADDR6 00000000 00000000 00000000 00000000			
0x00000054	MPUHn_EADDR6 00000000 00000000 00000000 00011111			
0x00000058	MPUHn_CTRL7 00000000 00000000 00000000 00000000			
0x0000005C	MPUHn_SADDR7 00000000 00000000 00000000 00000000			
0x00000060	MPUHn_EADDR7 00000000 00000000 00000000 00011111			
0x00000064	MPUHn_CTRL8 00000000 00000000 00000000 00000000			

Table 7-1. Memory layout of MPU AHB registers

Offset	+3	+2	+1	+0
0x00000068	MPUHn_SADDR8 00000000 00000000 00000000 00000000			
0x0000006C	MPUHn_EADDR8 00000000 00000000 00000000 00011111			
0x00000070	MPUHn_UNLOCK 00000000 00000000 00000000 00000000			
0x00000074	MPUHn_MID 00000000 00000000 00000000 00000000			

7.2.1 MPU AHB Control Register (MPUHn_CTRL0)

MPU AHB Control Register can be used by the software to configure the MPU AHB. This register provides enable bit to enable the MPU AHB monitoring and protection function. It also provides permission attributes for background region. It also provides controls for enabling or disabling of privileged mode overwrite feature and MPU STOP feature. Lastly it provides status of MPU AHB lock bit and Non-Maskable Interrupt flag.

MPU AHB Control Register (MPUHn_CTRL0)

Figure 7-2. MPU AHB Control Register (MPUHn_CTRL0)

MPUHn_CTRL0																													
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**
read0	read0	read0	read0	read0	AP[2]	AP[1]	AP[0]	read0	read0	read0	read0	read0	read0	MPUENC	MPUEN	read0	read0	read0	PROT	POEN	MPUSTOPEN	MPUSTOP	LST	read0	read0	read0	read0	read0	read0	NMICL	NMI
R0	R0	R0	R0	R0	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	R0WP	R	R0	R0	R0	RWP	RWP	RWP	R	R	R0	R0	R0	R0	R0	R0	R0WP1	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 7-2. MPU AHB Control Register (MPUHn_CTRL0) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26:24]	AP	Access Permissions for Background Region These bits are used to control access permissions for a background region. For more details about these bits, refer to Table 7-11. **Note:** AP[2:0] bits for background region can only be written when the MPU is disabled (MPUHn_CTRL0:MPUEN = '0').
[23:18]	read0	-
[17]	MPUENC	MPU AHB Enable Control '0': MPU AHB Monitoring and Protection Function is disabled '1': MPU AHB Monitoring and Protection Function is enabled Read returns '0'.
[16]	MPUEN	MPU AHB Enable Status '0': MPU AHB Monitoring and Protection Function is disabled. All accesses from the AHB Master Interface are passed on to the AHB memory interface without any protection '1': MPU AHB Monitoring and Protection Function is enabled

Table 7-2. MPU AHB Control Register (MPUHn_CTRL0) bits

Bit Position	Bit Field Name	Bit Description
[15:13]	read0	-
[12]	PROT	Privileged Mode Attribute When the privileged mode overwrite feature is available and POEN = '1', the privilege attribute on AHB memory interface is controlled by this bit. '0': Non-privileged mode '1': Privileged mode
[11]	POEN	Privileged Mode Overwrite Feature Enable '0': Privileged mode overwrite feature is disabled '1': Privileged mode overwrite feature is enabled **Note:** For availability of the privileged mode overwrite feature, refer to the device-specific datasheet.
[10]	MPUSTOPEN	Enable for MPU STOP Feature '0': MPU STOP feature is disabled '1': MPU STOP feature is enabled This bit together with MPU STOP input controls the STOP status of MPU AHB.
[9]	MPUSTOP	MPU STOP Status '0': MPU AHB is not in STOP mode '1': MPU AHB is in STOP mode (i.e. MPUSTOPEN = '1' and MPU STOP input is asserted). All accesses are blocked in this mode **Note:** For availability of the MPU STOP feature, refer to the device- specific datasheet.
[8]	LST	MPU Lock Status '0': MPU AHB is unlocked; registers in MPU AHB can be written '1': MPU AHB is locked; no registers (other than MPUHn_UNLOCK register) in MPU AHB can be written
[7:2]	read0	-
[1]	NMICL	NMI Interrupt Clear '0': No effect '1': Clears the NMI interrupt flag Read returns '0'.
[0]	NMI	NMI Interrupt Flag This interrupt flag indicates that the memory protection violation was detected for an AHB transfer.

Note: The register bits MPUHn_CTRL0:AP[2:0], MPUHn_CTRL0:MPUSTOPEN, MPUHm_CTRL0:POEN and MPUHn_CTRL0:PROT can only be written when the MPU is disabled (MPUHn_CTRL0:MPUEN = 0).

7.2.2 MPU AHB NMI Enable Register (MPUHn_NMIEN)

MPU AHB NMI Enable Register can be used by software to reset the NMI enable bit. The default value of the NMI enable bit is 1. This bit can be reset by software only once after a reset operation.

MPU AHB NMI Enable Register (MPUHn_NMIEN)

Figure 7-3. MPU AHB NMI Enable Register (MPUHn_NMIEN)

MPUHn_NMIEN																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMIEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 7-3. MPU AHB NMI Enable Register (MPUHn_NMIEN) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	NMIEN	NMI Interrupt Enable This bit decides whether the NMI interrupt flag is routed to an NMI interrupt signal or not. '0': NMI interrupt flag does not trigger an NMI interrupt signal 1': NMI interrupt flag triggers an NMI interrupt signal The value of this bit can be changed only once after reset.

7.2.3 MPU AHB Memory Error Control Register (MPUHn_MERRC)

This is a read-only register that provides the control information of AHB transfer for which memory protection violation was detected. This register can be read to get the privileged mode and transfer mode (write/read) information of the AHB transfer.

MPU AHB Memory Error Control Register (MPUHn_MERRC)

Figure 7-4. MPU AHB Memory Error Control Register (MPUHn_MERRC)

MPUHn_MERRC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	HPROT	HWRITE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-4. MPU AHB Memory Error Control Register (MPUHn_MERRC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	HPROT	AHB Transfer Privileged Mode. This bit provides the status of the HPROT signal of the AHB transfer for which memory protection violation is detected.
[0]	HWRITE	AHB Transfer Mode. This bit provides the status of the HWRITE signal of the AHB transfer for which memory protection violation is detected.

7.2.4 MPU AHB Memory Error Address Register (MPUHn_MERRA)

This is a read-only register that provides the address of AHB transfer for which memory protection violation was detected.

MPU AHB Memory Error Address Register (MPUHn_MERRA)

Figure 7-5. MPU AHB Memory Error Address Register (MPUHn_MERRA)

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

Table 7-5. MPU AHB Memory Error Address Register (MPUHn_MERRA) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	HADDR	AHB Address. The address of an AHB transfer for which memory protection violation was detected.

7.2.5 MPU AHB Region Control Registers (MPUHn_CTRL1~8)

MPU AHB Region Control Register is used to specify access permission for a particular region. Software can also use this register for enabling or disabling the particular region. MPUHn_CTRL1 Control Register for Region 1 is explained below.

Refer to the device-specific datasheet for the number of regions available.

MPU AHB Region Control Register for Region 1 (MPUHn_CTRL1)

Figure 7-6. MPU AHB Region Control Register for Region 1 (MPUHn_CTRL1)

MPUHn_CTRL1																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AP[2]	AP[1]	AP[0]	read0	read0	read0	read0	read0	read0	MPUENC	MPUEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	R0WP	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-6. MPU AHB Region Control Register for Region 1 (MPUHn_CTRL1) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10:8]	AP	Access Permissions These bits are used to control access permissions for region 1. For a detailed description of these bits refer to Table 7-11.
[7:2]	read0	-
[1]	MPUENC	Enable Control '0': Memory Protection for region 1 is disabled ' 1': Memory Protection for region 1 is enabled Read returns '0'. The enable status of the region can be read through the MPUHn_CTRL1:MPUEN bit.
[0]	MPUEN	Enable Status '0': Memory Protection for region 1 is disabled '1': Memory Protection for region 1 is enabled This is a read-only bit; writing to this bit has no effect.

Note:

For all eight MPU AHB Region Control registers, the Access Permission bits (i.e. MPUHn_CTRL1~8:AP) can only be written when the corresponding region is disabled (i.e. MPUHn_CTRL1~8:MPUEN = 0) or the MPU is disabled (MPUHn_CTRL0:MPUEN = 0). This also implies that when MPUHn_CTRL0:MPUEN = 1 and MPUHn_CTRL1~8:MPUEN = 1:

- 32- or 16-bit access to this register is not allowed
- 8-bit access is required to disable the MPU

7.2.6 MPU AHB Start Address Registers (MPUHn_SADDR1~8)

Each region in MPU AHB can be defined by specifying the start address and end address for that particular region. The MPUHn_SADDR1~8 registers are used to specify the start address for the eight regions. The start address indicates the first address for that region. MPUHn_SADDR1 is the Start Address Register for Region 1 and is explained below. Refer to the device-specific datasheet for the number of regions available.

MPU AHB Start Address Register for Region 1 (MPUHn_SADDR1)

Figure 7-7. MPU AHB Start Address Register for Region 1 (MPUHn_SADDR1)

MPUHn_SADDR1																															
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**
SADDR[31]	SADDR[30]	SADDR[29]	SADDR[28]	SADDR[27]	SADDR[26]	SADDR[25]	SADDR[24]	SADDR[23]	SADDR[22]	SADDR[21]	SADDR[20]	SADDR[19]	SADDR[18]	SADDR[17]	SADDR[16]	SADDR[15]	SADDR[14]	SADDR[13]	SADDR[12]	SADDR[11]	SADDR[10]	SADDR[9]	SADDR[8]	SADDR[7]	SADDR[6]	SADDR[5]	SADDR[4]	SADDR[3]	SADDR[2]	SADDR[1]	SADDR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-7. MPU AHB Start Address Register for Region 1 (MPUHn_SADDR1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SADDR	Start Address Start address for region 1

Note: the MPUHn_SADDR1~8 registers can only be written when the corresponding region is disabled (MPUHm_CTRL1~8:MPUEN = 0) or the MPU is disabled (MPUHn_CTRL0:MPUEN = 0).

7.2.7 MPU AHB End Address Registers (MPUHn_EADDR1~8)

Each region in MPU AHB can be defined by specifying a start address and end address for that particular region. MPUHn_EADDR1~8 registers are used to specify the end addresses for the eight regions. The end address indicates the last address for that region. The MPUHn_EADDR1 End Address Register for Region 1 is explained below. Refer to the device-specific datasheet for the number of regions available.

MPU AHB End Address Register for Region 1 (MPUHn_EADDR1)

Figure 7-8. MPU AHB End Address Register for Region 1 (MPUHn_EADDR1)

MPUHn_EADDR1																															
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**
EADDR[31]	EADDR[30]	EADDR[29]	EADDR[28]	EADDR[27]	EADDR[26]	EADDR[25]	EADDR[24]	EADDR[23]	EADDR[22]	EADDR[21]	EADDR[20]	EADDR[19]	EADDR[18]	EADDR[17]	EADDR[16]	EADDR[15]	EADDR[14]	EADDR[13]	EADDR[12]	EADDR[11]	EADDR[10]	EADDR[9]	EADDR[8]	EADDR[7]	EADDR[6]	EADDR[5]	EADDR[4]	EADDR[3]	EADDR[2]	EADDR[1]	EADDR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	R1	R1	R1	R1	R1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Table 7-8. MPU AHB End Address Register for Region 1 (MPUHm_EADDR1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EADDR	End Address End address for region 1

Note: MPUHn_EADDR1~8 registers can only be written when the corresponding region is disabled (MPUHn_CTRL1~8:MPUEN = 0) or the MPU is disabled (MPUHn_CTRL0:MPUEN = 0).

7.2.8 MPU AHB Unlock Register (MPUHn_UNLOCK)

The software can use this register to lock or unlock the MPU AHB registers for write access.

MPU AHB Unlock Register (MPUHn_UNLOCK)

Figure 7-9. MPU AHB Unlock Register (MPUHn_UNLOCK)

MPUHn_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-9. MPU AHB Unlock Register (MPUHn_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	MPU AHB Unlock The MPU AHB Unlock Register protects the MPU AHB module from being modified accidently by software. The MPU AHB registers cannot be written until this register has been written with a specific unlock value. The correct value for unlocking can be written only in privileged mode. Reading this register always returns '0'. To lock the MPU AHB again software must write another value specific to lock. A write access to the MPU AHB registers without unlocking or writing value other than the lock or unlock value to this register causes a protection error. **Notes:** 1. This register cannot be written by an 8- or 16-bit write access; any such access causes a protection error. 2. For more details on the lock and unlock values, refer to the device-specific datasheet.

7.2.9 MPU AHB Module ID Register (MPUHn_MID)

This is a read-only register with a unique module identification number which identifies the version of the MPU AHB module used in the MCU. Refer to the device-specific datasheet for the module identification number of the MPU AHB module.

MPU AHB Module ID Register (MPUHn_MID)

Figure 7-10. MPU AHB Module ID Register (MPUHn_MID)

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

Table 7-10. MPU AHB Module ID Register (MPUHn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID The MPU AHB module implemented in the device may vary from device to device. This register identifies the particular version of the hardware used in the device. This register helps in developing the software to hardware version implemented in the device **Note:** For more details refer to the device-specific datasheet for the Module ID number.

7.3 Operation of the MPU AHB

This section describes the operation of the MPU AHB.

MPU AHB region granularity and priority decision

The MPU AHB supports up to 8 regions and provides the start and end addresses for each of these eight regions. MPU AHB regions are defined with a granularity of 32 bytes.

Each region is identified as Region 1, Region 2, Region 3 and so on up to Region 8, which is given highest priority. Region 7 has second highest priority and so on down to Region 1, which has the second lowest priority. The Background Region is designated the lowest priority region.

The Start Address specifies the first address of the region and is specified by registers MPUHn_SADDR1 to MPUHn_SADDR8 for region 1 to region 8 respectively. Since the region granularity is 32 bytes, the least significant 5 bits of the start addresses will read 0.

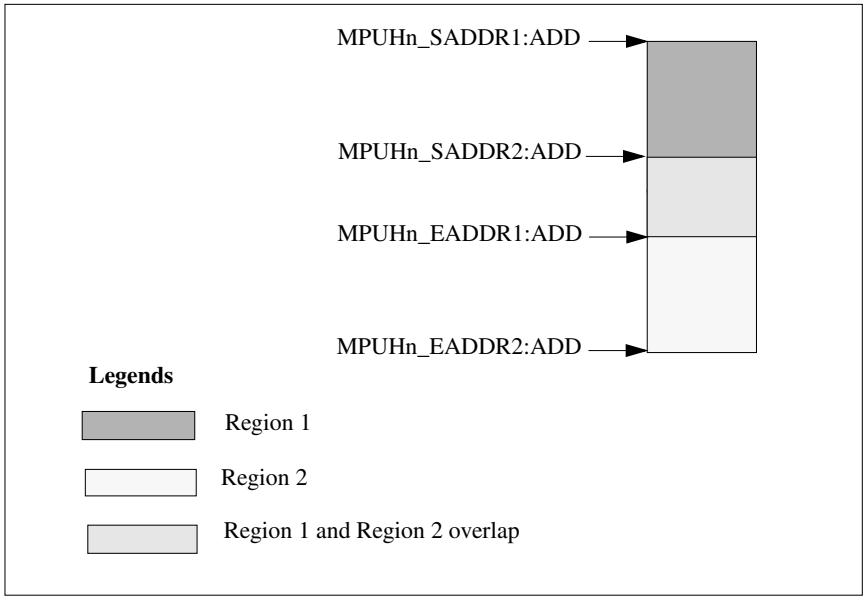
End Addresses are specified by registers MPUHn_EADDR1 to MPUHn_EADDR8n for region 1 to region 8 respectively. The least significant 5 bits of the End Address registers are read only bits and will always read 1. This ensures the granularity of 32 bytes.

In the MPU AHB, regions may overlap each other, as shown in Figure 7-11. Hence it is also possible that an AHB address for any transfer may match with multiple regions. Whenever the AHB Transfer Address matches with multiple regions, the permissions corresponding to the highest priority region (among all matching regions) is applied.

AHB burst monitoring

The MPU AHB only supports Single Transfer Monitoring and not AHB Burst Monitoring. Therefore, it can only be used together with AHB Masters that do not request Burst transfers.

Figure 7-11. Example region overlapping



MPU access permissions

Region Control registers in the MPU AHB (i.e. MPUHn_CTRL1 to MPUHn_CTRL8) are used to control the access permission to Regions 1 to 8. Also MPUHn_CTRL0 is used to control the access permission for the Background Region.

Table 7-11 describes Access Permission bits (AP) in these registers and corresponding permission attributes.

Table 7-11. Access permissions

AP bits	Access in privileged mode	Access in non- privileged mode	Comment
000 (default)	No access	No access	All bus accesses are blocked and hence would generate a memory protection violation.
001	Read, write	No access	Reads and writes are permitted in privileged mode only. Any access in non- privileged mode would generate a memory protection violation.
010	Read, write	Read only	Reads and writes are permitted in privileged mode. Only reads are permitted in non-privileged mode. Writes in non- privileged mode would generate a memory protection violation.
011	Read, write	Read, write	All bus transfers are permitted. No memory protection violation is generated in this mode.
100	No access	No access	All bus accesses are blocked and hence generate a memory protection violation.
101	Read only	No access	Reads are permitted in privileged mode only. Writes in privileged mode and any access in non-privileged mode would generate a memory protection violation.
110	Read only	Read only	Reads are permitted in privileged as well as non-privileged mode. Any write access would generate a memory protection violation.
111	Read, write	Read, write	All bus transfers are permitted. No memory protection violation is generated in this mode.

Bus monitor and protection logic

All transfers on the AHB Master Interface are monitored and checked for permitted access.

- Bus monitor and protection logic within the MPU AHB compares the address of the current transfer with the start and end addresses of each region to find a region match, i.e. where the current transfer matches one of the eight defined regions
- As explained in MPU AHB region granularity and priority decision, the AHB transfer address may match multiple regions where the permission attributes of the region with highest priority are checked against the attributes of the currently applied transfer from the AHB master
- If the attributes of the currently applied transfer are within the permitted attributes, the current transaction is passed on to the AHB memory interface
- If the attributes are not within permitted attributes, the current transfer is blocked. The Non- Maskable Interrupt flag (MPUHn_CTRL0:NMI) is set. The address and control information of the current transfer is stored in MPUHn_MERRA and MPUHn_MERRC respectively
- All further transfers are blocked until the MPUHn_CTRL0:NMI flag is cleared by software. Further monitoring of the AHB transfer addresses is also stalled until the MPUHn_CTRL0:NMI flag is cleared.

When a transfer is blocked, the MPU AHB performs the following actions:

- It drives idle transfers on the AHB memory interface
- It generates an error response on the AHB master interface

MPU STOP feature

Optionally the MPU AHB supports an MPU STOP feature.

When this mode is enabled, all accesses to memory space are blocked and the MPU AHB performs the following actions:

- It drives idle transfers on the AHB memory interface

- It generates an error response on the AHB master interface

For more details on the MPU STOP related register bits refer to Figure 7-2.

Privileged mode overwrite feature

Optionally the MPU AHB supports privileged mode overwrite feature.

When this mode is enabled, the privileged mode attribute on the AHB memory interface is set by setting the MPUHn_CTRL0:PROT bit as explained in 7.2.1 MPU AHB Control Register (MPUHn_CTRL0).

Note: Bus monitor and protection logic for detecting memory protection violation uses the privileged mode attribute on the AHB master interface and not the MPUHn_CTRL0:PROT bit, even when the privileged mode overwrite feature is enabled.

7.4 Notes on using the MPU AHB

This section is the ‘Programmer’s Guide’, which lists the usage notes for programming the MPU AHB module. Please read these guidelines before programming the module.

General usage notes

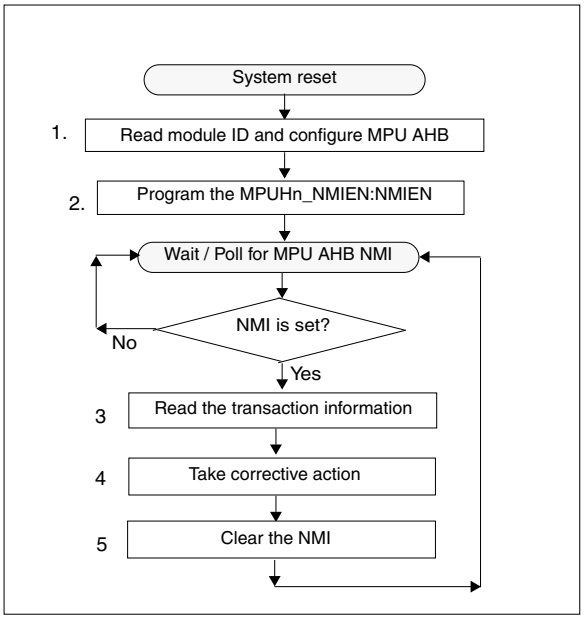
- Reserved bits return undefined values. The software programs shall be independent of the values read from the reserved register bits
- The MPU AHB supports storage of information of the first memory protection violation only on the bus. Therefore, once an NMI interrupt flag is set, further monitoring of the AHB master interface is stalled until the MPUHn_CTRL0:NMI bit is cleared. This implies that any memory protection violation occurring on the bus while the MPUHn_CTRL0:NMI is set is simply ignored by the MPU AHB

Note: Software developers must therefore try to keep the ISR size small so that the NMI interrupt of the MPU AHB does not remain unattended for long.

Steps in programming the MPU AHB module

Figure 7-12 gives the general steps a programmer shall follow while using the MPU AHB module.

Figure 7-12. Programmer’s flowchart



1. After the system reset, the software detects the module ID number of the MPU AHB by reading the MPUHn_MID register. This helps it in identifying the attributes and capabilities supported by the MPU AHB module. The software then configures the MPU AHB by setting the appropriate registers.
2. By default, the MPU AHB propagates the MPUHn_CTRL0:NMI flag to the CPU through the Interrupt Controller. If polling mode is desired, the software can reset the MPUHn_NMIEN:NMIEN bit to ‘0’.

Note: The MPUHm_NMIEN:NMIEN can be written only once after reset. Subsequent write accesses to this bit have no visible impact on the state of this bit.

3. When the NMI is triggered (or in polling mode if the software detects during its polling cycle that the MPUHn_CTRL0:NMI status flag is set), the CPU is invoked and reads the status information collected and stored by the MPU AHB in its CSR.
4. The software diagnoses the information about the protection violation transaction and initiates a corrective action (if any).

5. Once the software has processed the information from the status registers, it shall clear the MPUHn_CTRL0:NMI flag by writing “1” to the MPUHn_CTRL0:NMICL bit. Clearing the MPUHn_CTRL0:NMI flag ensures that the MPU AHB starts monitoring the AHB Master Interface again to check for memory protection violation.

Note: Software may clear the NMI flag before taking corrective action; therefore steps 4 and 5 could be interchanged.

8. Programmable CRC

This chapter explains the function and operation of the Programmable CRC.

8.1 Outline of the Programmable CRC

This section describes the features and the block diagram of the Programmable CRC.

Features of Programmable CRC

The Programmable CRC is a software configurable module with serial CRC calculation logic as hardware implementation. The serial CRC logic works on modulo-2 arithmetic for calculation of checksum. The CRC module can detect errors in data blocks by calculating a checksum.

- Programmable 8-, 16-, 24-, or 32-bit input data width
- Programmable polynomial value (polynomial degree from 2 to 32)
- Programmable initial seed value
- Programmable final checksum XOR value
- Interrupt and DMA trigger capability
- Configurable input/output bit reflection and byte swapping. This facilitates the different settings of common CRC standards and the handling of data organized in little or big endian format
- Supports PPU
- Supports block/multiple data transfers (more than 32-bit)

Areas of application

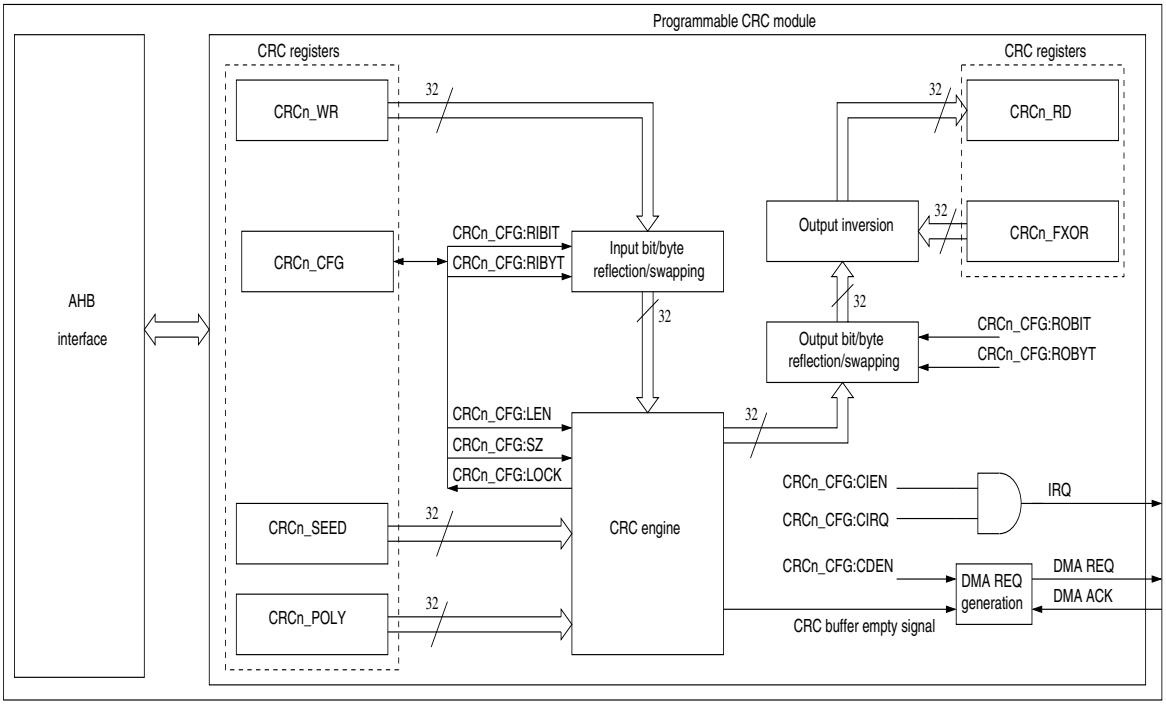
- Data security/integrity
- Communication protocols

Programmable CRC module can be configured to widely used common CRC standards, some of them are listed below:

- CRC-32-IEEE 802.3
- CRC-16-CCITT
- CRC-8-CCITT
- CRC-5-USB
- CRC-XMODEM
- 12-bit CRC
- 10-bit CRC
- 8-bit CRC

Block diagram of Programmable CRC

Figure 8-1. Block diagram of Programmable CRC



8.2 Programmable CRC registers

All Programmable CRC registers are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of Programmable CRC

The following registers are available for each instance of Programmable CRC:

- CRC Polynomial Register (CRCn_POLY)
- CRC Seed Register (CRCn_SEED)
- CRC Final XOR Register (CRCn_FXOR)
- CRC Configuration Register (CRCn_CFG)
- CRC Write Register (CRCn_WR)
- CRC Read Register (CRCn_RD)

Memory layout of Programmable CRC registers

Table 8-1. Memory layout of Programmable CRC registers

Offset	+3	+2	+1	+0
0x00000000	CRCn_POLY 00000100 11000001 00011101 10110111			
0x00000004	CRCn_SEED 11111111 11111111 11111111 11111111			
0x00000008	CRCn_FXOR 11111111 11111111 11111111 11111111			
0x0000000C	CRCn_CFG 00000000 11100000 00000000 00000000			
0x00000010	CRCn_WR 00000000 00000000 00000000 00000000			
0x00000014	CRCn_RD 00000000 00000000 00000000 00000000			

8.2.1 CRC Polynomial Register (CRCn_POLY)

The CRC Polynomial Register (CRCn_POLY) defines the polynomial value for the CRC checksum calculation.

CRC Polynomial Register (CRCn_POLY)

Figure 8-2. CRC Polynomial Register (CRCn_POLY)

CRCn_POLY																															
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**
POLY[31]	POLY[30]	POLY[29]	POLY[28]	POLY[27]	POLY[26]	POLY[25]	POLY[24]	POLY[23]	POLY[22]	POLY[21]	POLY[20]	POLY[19]	POLY[18]	POLY[17]	POLY[16]	POLY[15]	POLY[14]	POLY[13]	POLY[12]	POLY[11]	POLY[10]	POLY[9]	POLY[8]	POLY[7]	POLY[6]	POLY[5]	POLY[4]	POLY[3]	POLY[2]	POLY[1]	POLY[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Table 8-2. Polynomial Register (CRCn_POLY) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	POLY	CRC Polynomial The CRCn_POLY contains the CRC polynomial. The degree of the polynomial must be between 2 to 32. Initial value is the common CRC-32 polynomial 0x04C11DB7 ($x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$). The highest degree of coefficient should be used to configure polynomial/checksum length (CRCn_CFG:LEN).

Note: If the polynomial length as defined by CRCn_CFG:LEN is less than 32-bit, then the upper bits [31:LEN] must be written to '0' by the programmer. The highest order degree must not be set to '1' while configuring CRCn_POLY register, as it is implicitly defined by CRCn_CFG:LEN.

8.2.2 CRC Seed Register (CRCn_SEED)

The CRC Seed Register (CRCn_SEED) defines the initial value for the CRC checksum calculation.

CRC Seed Register (CRCn_SEED)

Figure 8-3. CRC Seed Register (CRCn_SEED)

CRCn_SEED																															
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**
SEED[31]	SEED[30]	SEED[29]	SEED[28]	SEED[27]	SEED[26]	SEED[25]	SEED[24]	SEED[23]	SEED[22]	SEED[21]	SEED[20]	SEED[19]	SEED[18]	SEED[17]	SEED[16]	SEED[15]	SEED[14]	SEED[13]	SEED[12]	SEED[11]	SEED[10]	SEED[9]	SEED[8]	SEED[7]	SEED[6]	SEED[5]	SEED[4]	SEED[3]	SEED[2]	SEED[1]	SEED[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 8-3. CRC Seed Register (CRCn_SEED) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SEED	CRC SEED CRCn_SEED contains the initial value for checksum calculation. If the seed value is not initialized for the next operation (even if the seed value is identical to the previous operation), then calculation is continued from the last state with the current checksum value as initial value. Initial value for seed register is 0xFFFFFFFF.

Note: The CRCn_SEED register should be configured with respect to the polynomial length (CRCn_CFG:LEN). If the polynomial length is less than 32-bit, then the upper bits [31:LEN] must be written to '0' by the programmer.

8.2.3 CRC Final XOR Register (CRCn_FXOR)

The CRC Final XOR register (CRCn_FXOR) contains the values to be XOR'ed with the preliminary checksum to finalize the CRC calculation.

CRC Final XOR Register (CRCn_FXOR)

Figure 8-4. CRC Final XOR Register (CRCn_FXOR)

CRCn_FXOR																															
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**
FXOR[31]	FXOR[30]	FXOR[29]	FXOR[28]	FXOR[27]	FXOR[26]	FXOR[25]	FXOR[24]	FXOR[23]	FXOR[22]	FXOR[21]	FXOR[20]	FXOR[19]	FXOR[18]	FXOR[17]	FXOR[16]	FXOR[15]	FXOR[14]	FXOR[13]	FXOR[12]	FXOR[11]	FXOR[10]	FXOR[9]	FXOR[8]	FXOR[7]	FXOR[6]	FXOR[5]	FXOR[4]	FXOR[3]	FXOR[2]	FXOR[1]	FXOR[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 8-4. CRC Final XOR Register (CRCn_FXOR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FXOR	CRC XOR Data The contents of CRCn_FXOR register are XOR'ed with the preliminary checksum data (after CRCn_CFG:ROBIT/ROBYT settings have been applied) and then the final checksum is moved to CRCn_RD register. Initial value for final XOR register is 0xFFFFFFFF.

Note: The bits of this register affect the corresponding bits of the CRCn_RD register. Therefore, the bits not belonging to the checksum should be written to '0'. For the position of the checksum bits depending on the used output bit/byte reflection refer to Table 8-10.

8.2.4 CRC Configuration Register (CRCn_CFG)

The CRC Configuration Register (CRCn_CFG) is used to set the operation mode of the CRC module. CRCn_CFG register describes the polynomial/checksum length, input data size, and input/output bit/byte reflection. It indicates the status of CRC operation. Interrupt and DMA requests are also configured using CRCn_CFG register.

CRC Configuration Register (CRCn_CFG)

Figure 8-5. CRC Configuration Register (CRCn_CFG)

CRCn_CFG																															
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**
read0	read0	read0	LOCK	read0	CDEN	CIEN	CIRQ	SZ[1]	SZ[0]	LEN[5]	LEN[4]	LEN[3]	LEN[2]	LEN[1]	LEN[0]	read0	read0	read0	read0	RIBIT	RIBYT	ROBIT	ROBYT	read0	read0	read0	read0	read0	read0	read0	CIRQCLR
Rp0	Rp0	Rp0	Rp	Rp0	RpWp	RpWp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1
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	0	0	0	0	0	0	0

Table 8-5. CRC Configuration Register (CRCn_CFG) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28]	LOCK	CRC Engine Status bit This bit indicates the status of the CRC engine. '0': CRC engine is ready, new data can be written to CRC registers '1': CRC engine is busy and writing to CRC registers is not possible. If the data is written to the CRC registers when LOCK bit is '1', then an error response is generated
[27]	read0	-
[26]	CDEN	DMA Request Enable bit This bit enables/disables the DMA request. '0': Disable the DMA request '1': Enable the DMA request DMA request is generated when CRC is in buffer empty state and CRCn_CFG:CDEN is set. DMA ISR should clear this bit after final transfer. Clearing this bit by CPU at any other time might lead to unwanted behavior.

Table 8-5. CRC Configuration Register (CRCn_CFG) bits

Bit Position	Bit Field Name	Bit Description
[25]	CIEN	CRC Interrupt Enable bit to CPU This bit enables/disables the interrupt request. '0': Disable the interrupt request '1': Enable the interrupt request The IRQ is triggered when CRCn_CFG:CIEN bit is enabled and CRC interrupt flag (CRCn_CFG:CIRQ) is set.
[24]	CIRQ	CRC Interrupt Flag '0': No interrupt request **Note:** CPU clears interrupt flag by writing CRCn_CFG:CIRQCLR bit to '1' This bit indicates the interrupt status of CRC. '1': Interrupt request, Checksum has been calculated by CRC engine and is available in CRCn_RD register.
[23:22]	SZ	CRC Input Data Size Configuration bits These bits are used to configure the input data size as follows: '00': 8-bit '01': 16-bit '10': 24-bit '11': 32-bit
[21:16]	LEN	CRC Polynomial/Checksum Length Configuration bits These bits are used to configure the length (degree) of CRC polynomial/checksum as follows: '100000':32 '011111':31 .... '000010':2 **Note:** The following settings are not supported: CRCn_CFG:LEN > 32 CRCn_CFG:LEN < 2
[15:12]	read0	-
[11]	RIBIT	Reflect Input Bits '0': Disable input bit reflection '1': Enable input bit reflection When the input data in the CRCn_WR register is passed to the CRC engine, the bit ordering of each byte within input data is reversed. For more details refer to 8.3 Operation of the Programmable CRC.

Table 8-5. CRC Configuration Register (CRCn_CFG) bits

Bit Position	Bit Field Name	Bit Description
[10]	RIBYT	Reflect Input Bytes '0': Disable input byte reflection (swapping) '1': Enable input byte reflection (swapping) When the input data in the CRCn_WR register is passed to the CRC engine, the byte ordering of input data is reversed. Only the bytes of the configured input data size CRCn_CFG:SZ are affected. For more details refer to 8.3 Operation of the Programmable CRC. **Note:** For 8-bit input data, this setting has no effect.
[9]	ROBIT	Reflect Output Bits '0': Disable output bit reflection '1': Enable output bit reflection The bit ordering of each byte within checksum is reversed, before passing the checksum to final XOR'ing stage. For more details refer to 8.3 Operation of the Programmable CRC.
[8]	ROBYT	Reflect Output Bytes '0': Disable output byte reflection (swapping) '1': Enable output byte reflection (swapping) The byte ordering of checksum data is reversed, before passing the byte aligned polynomial length of checksum data to final XOR'ing stage. For more details refer to 8.3 Operation of the Programmable CRC. **Note:** For the checksum data less than or equal to 8 bits, this setting has no effect.
[7:1]	read0	-
[0]	CIRQCLR	Interrupt Clear This bit clears the CRC interrupt flag. '0': Write '0' is ignored, reading this bit always returns '0' '1': Clear CRC interrupt flag (CRCn_CFG:CIRQ)

8.2.5 CRC Write Register (CRCn_WR)

The input data for the CRC checksum calculation must be written to the CRC Write Register (CRCn_WR).

CRC Write Register (CRCn_WR)

Figure 8-6. CRC Write Register (CRCn_WR)

CRCn_WR																															
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**
CRCWR[31]	CRCWR[30]	CRCWR[29]	CRCWR[28]	CRCWR[27]	CRCWR[26]	CRCWR[25]	CRCWR[24]	CRCWR[23]	CRCWR[22]	CRCWR[21]	CRCWR[20]	CRCWR[19]	CRCWR[18]	CRCWR[17]	CRCWR[16]	CRCWR[15]	CRCWR[14]	CRCWR[13]	CRCWR[12]	CRCWR[11]	CRCWR[10]	CRCWR[9]	CRCWR[8]	CRCWR[7]	CRCWR[6]	CRCWR[5]	CRCWR[4]	CRCWR[3]	CRCWR[2]	CRCWR[1]	CRCWR[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-6. CRC Write Register (CRCn_WR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CRCWR	CRC Write Register The CRCn_WR register contains the input data, for which the CRC checksum is to be calculated. Writing to this register starts the CRC calculation process. After pre-processing (bit/byte reflection/swapping) the contents of the CRCn_WR register are passed to the CRC engine (the contents of CRCn_SEED register are also provided). There it is divided by the content of CRCn_POLY register in modulo-2 arithmetic to get the final checksum after post-processing (bit/byte reflection/swapping and XOR'ing).

Note: The size of input data is configured by CRCn_CFG:SZ, where 8, 16, 24, and 32 bits are only supported as data size. If the input data size is less than 32-bit (i.e. 8, 16, or 24 bits), then the invalid/unused bits are considered as don't care (X).

8.2.6 CRC Read Register (CRCn_RD)

The CRC Read Register (CRCn_RD) contains the final checksum of the data written to the CRCn_WR register.

CRC Read Register (CRCn_RD)

Figure 8-7. CRC Read Register (CRCn_RD)

CRCn_RD																															
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**
CRCRD[31]	CRCRD[30]	CRCRD[29]	CRCRD[28]	CRCRD[27]	CRCRD[26]	CRCRD[25]	CRCRD[24]	CRCRD[23]	CRCRD[22]	CRCRD[21]	CRCRD[20]	CRCRD[19]	CRCRD[18]	CRCRD[17]	CRCRD[16]	CRCRD[15]	CRCRD[14]	CRCRD[13]	CRCRD[12]	CRCRD[11]	CRCRD[10]	CRCRD[9]	CRCRD[8]	CRCRD[7]	CRCRD[6]	CRCRD[5]	CRCRD[4]	CRCRD[3]	CRCRD[2]	CRCRD[1]	CRCRD[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-7. CRC Read Register (CRCn_RD) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CRCRD	CRC Read Data The CRC Read Register contains the result (final checksum) of the CRC calculation after post-processing (applying CRCn_CFG:ROBIT, CRCn_CFG:ROBYT, and CRCn_FXOR settings). Writing any value on CRCn_RD register changes its content and it does not affect CRC calculation.

Note: The polynomial length (CRCn_CFG:LEN) provides the length of the final checksum. If the polynomial length is less than 32-bit, then bits not belonging to the checksum are '0' (in case the invalid bits of the CRCn_FXOR register are not programmed to'0', then these bits in CRCn_RD register might also be'1'). The bit/byte reflection settings (CRCn_CFG:ROBIT/ROBYT) can influence the checksum in CRCn_RD register. For the position of the checksum bits refer to Table 8-10.

8.3 Operation of the Programmable CRC

This section describes operation of Programmable CRC in detail.

For more details on flowcharts for CRC operation see 8.3.1 CRC operation flowcharts, on CRC calculation flow see 8.3.2 CRC input data and checksum calculation flow, and on an example for CRC calculation see 8.3.3 CRC calculation example.

8.3.1 CRC operation flowcharts

The flowcharts Figure 8-8, Figure 8-9, and Figure 8-10 show the steps to configure CRC registers and to perform a CRC calculation.

Figure 8-8. Polling based CRC operation

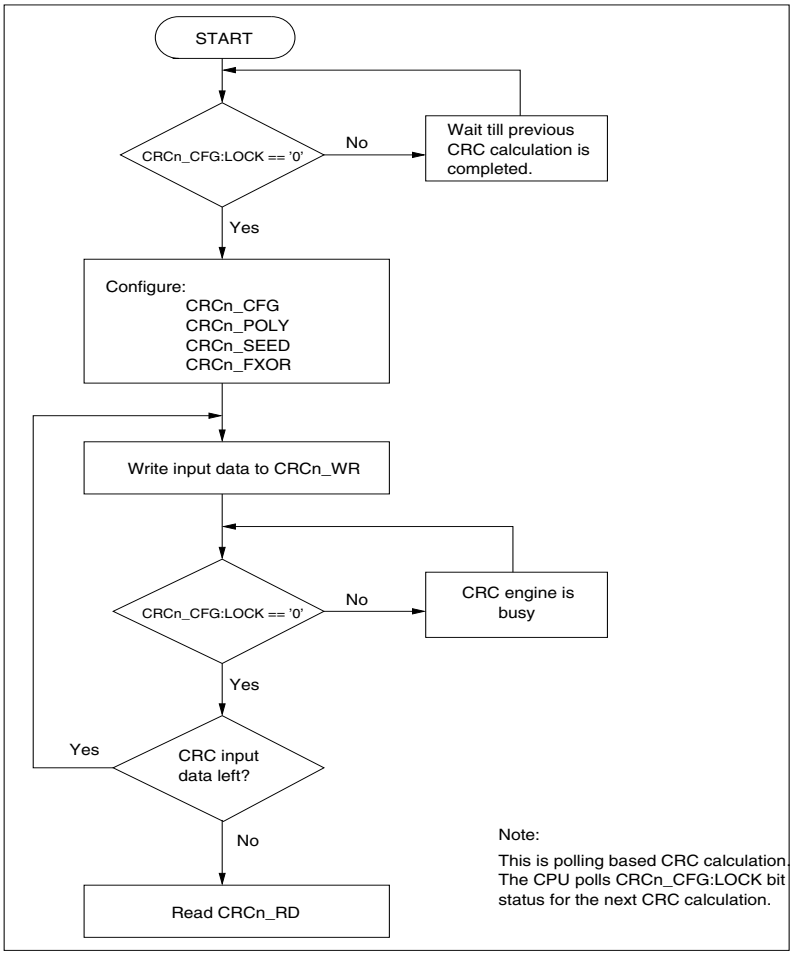


Figure 8-9. CRC operation with IRQ

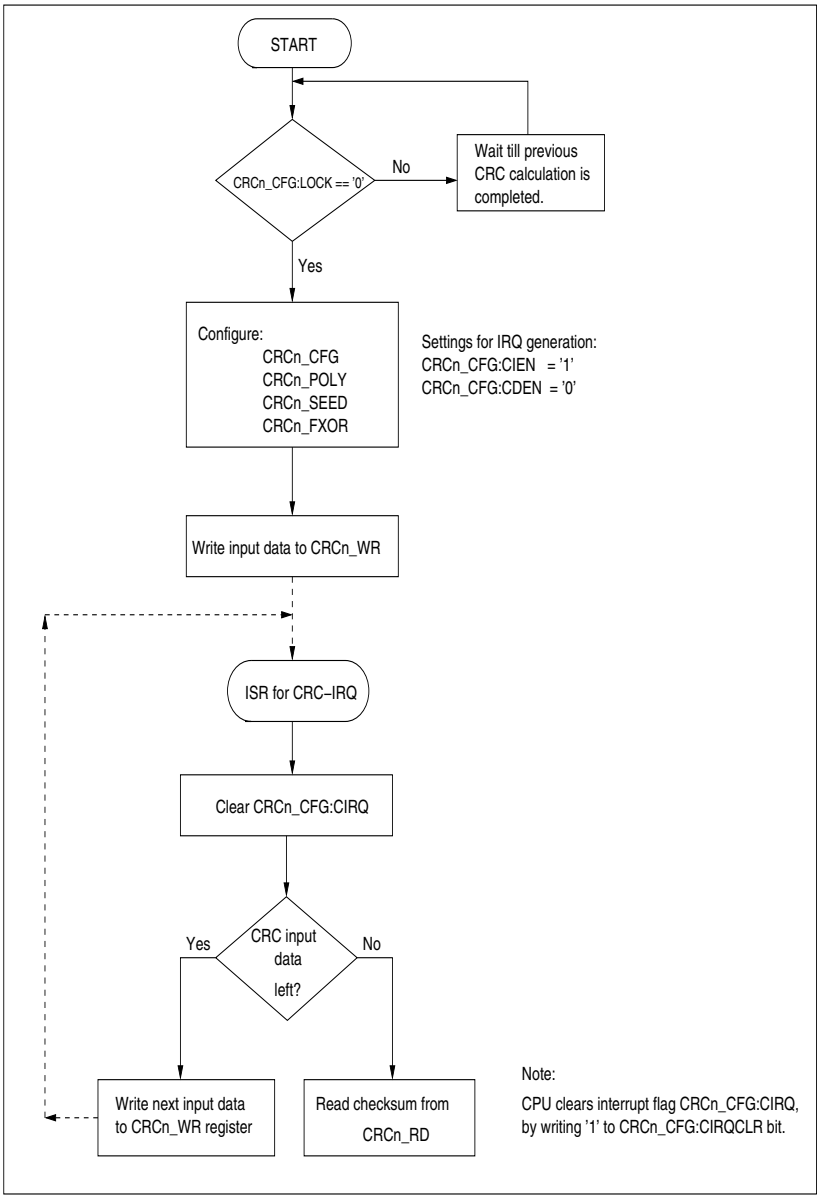
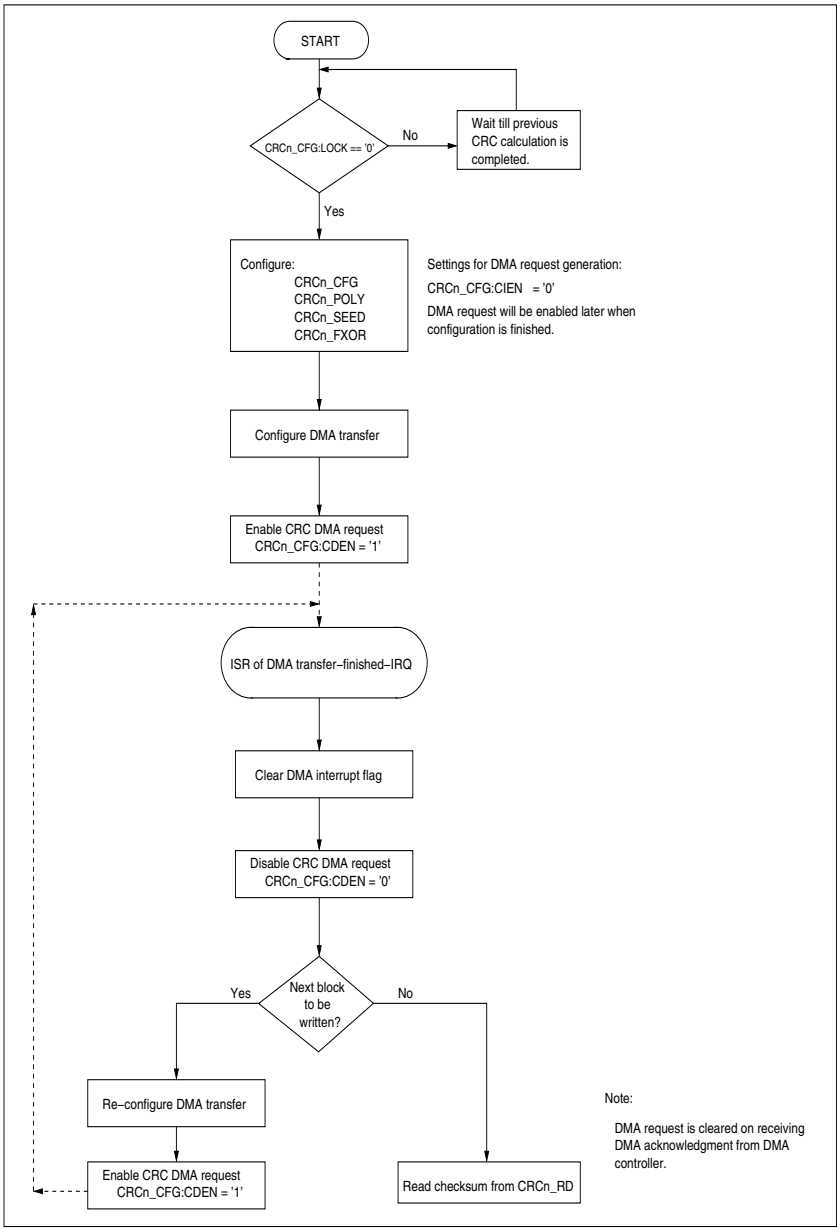
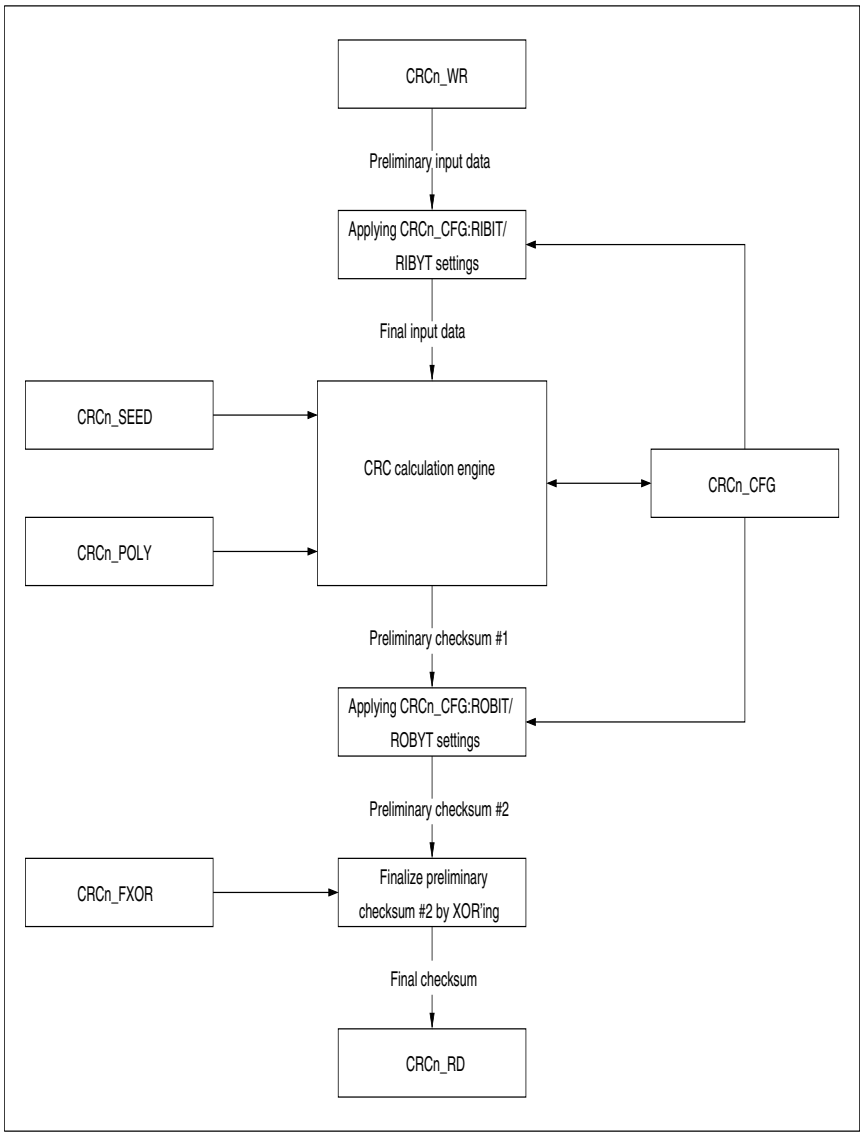


Figure 8-10. CRC operation with DMA request



8.3.2 CRC input data and checksum calculation flow

Figure 8-11. Block diagram of CRC input data and checksum calculation flow



1. The input data for which CRC is to be calculated is written to CRCn_WR register. This is the 'preliminary input data'.
2. The 'preliminary input data' bytes can be swapped/reflected bit-wise using CRCn_CFG:ROBIT and/or byte-wise using CRCn_CFG:ROBYT before they enter the CRC engine. The settings are shown below:

   'preliminary input data' in CRCn_WR register:

   A7----A0 B7----B0 C7----C0 D7----D0

   If the input data size is less than 32-bit (SZ < ‘11’), then the remaining bits (8-,16-, or 24-bit) of the data are considered as don’t care (X) as shown in below table.

Table 8-8. Preliminary input data bit-wise and/or byte-wise reflection/swapping

RIBYT	RIBIT	SZ	Final input data for CRC engine			
			+3	+2	+1	+0
‘0’	‘0’	‘00’	XXXX XXXX	XXXX XXXX	XXXX XXXX	D7------- D0
		‘01’	XXXX XXXX	XXXX XXXX	C7--------C0	D7--------D0
		‘10’	XXXX XXXX	B7--------B0	C7--------C0	D7--------D0
		‘11’	A7--------A0	B7--------B0	C7--------C0	D7--------D0
	‘1’	‘00’	XXXX XXXX	XXXX XXXX	XXXX XXXX	D0--------D7
		‘01’	XXXX XXXX	XXXX XXXX	C0--------C7	D0--------D7
		‘10’	XXXX XXXX	B0--------B7	C0--------C7	D0--------D7
		‘11’	A0---------A7	B0--------B7	C0--------C7	D0--------D7
‘1’	‘0’	‘00’	XXXX XXXX	XXXX XXXX	XXXX XXXX	D7--------D0
		‘01’	XXXX XXXX	XXXX XXXX	D7--------D0	C7--------C0
		‘10’	XXXX XXXX	D7--------D0	C7--------C0	B7--------B0
		‘11’	D7---------D0	C7--------C0	B7--------B0	A7--------A0
	‘1’	‘00’	XXXX XXXX	XXXX XXXX	XXXX XXXX	D0--------D7
		‘01’	XXXX XXXX	XXXX XXXX	D0--------D7	C0--------C7
		‘10’	XXXX XXXX	D0--------D7	C0--------C7	B0--------B7
		‘11’	D0---------D7	C0--------C7	B0--------B7	A0--------A7

3. The 'preliminary input data' after applying the settings of CRCn_CFG:RIBIT/RIBYT results in the 'final input data', which is sent to the CRC engine for checksum calculation.
4. The CRCn_SEED register provides the initial value to the CRC engine. The required polynomial is provided by CRCn_POLY register. The CRC engine starts its operation once CRCn_WR register is written with the input data.
5. CRC engine performance: The performance of CRC engine for CRC checksum calculation is based on the input data size and number of clock cycles (bus clock) required to complete a calculation. The Table 8-9 shows number of clock cycles required to get final checksum at CRCn_RD register with respect to input data size.

Table 8-9. Clock cycles requirement for checksum calculation

Input data size	Number of clock cycles required for final checksum at CRCn_RD
8-bit	Input data size (8-bit) + 2 = 10 clock cycles.
16-bit	Input data size (16-bit) + 2 = 18 clock cycles.
24-bit	Input data size (24-bit) + 2 = 26 clock cycles.

Table 8-9. Clock cycles requirement for checksum calculation

Input data size	Number of clock cycles required for final checksum at CRCn_RD
32-bit	Input data size (32-bit) + 2 = 34 clock cycles.

6. The 'preliminary checksum #1' bytes can be swapped/reflected bit-wise using CRCn_CFG:ROBIT and/or byte-wise using CRCn_CFG:ROBYT. Table 8-10 shows at which positions the checksum bits of 'preliminary checksum #1' S[(LEN-1):0] will be located in 'preliminary checksum #2' after CRCn_CFG:ROBIT/ROBYT settings have been applied. Only some examples of different CRCn_CFG:LEN configurations are shown.

Note: Only some examples for CRCn_CFG:LEN are shown.

Table 8-10. Preliminary checksum #1 bit-wise and/or byte-wise reflection/swapping

ROBYT	ROBIT	LEN	Preliminary checksum #2				
			+3	+2	+1	+0	Action
‘0’	‘0’	32	S31---S24	S23---S16	S15---S8	S7---S0	No swapping/reflection. The checksum is aligned with the polynomial degree/length.
		21	‘0000 0000’	‘000 S20---S16’	S15---S8	S7---S0	No swapping/reflection. The checksum is aligned with the polynomial degree/length. The bits S21 to S31 are ’0’.
		16	‘0000 0000’	‘0000 0000’	S15---S8	S7---S0	No swapping/reflection. The checksum is aligned with the polynomial degree/length. The bits S16 to S31 are ’0’.
		3	‘0000 0000’	‘0000 0000’	‘0000 0000’	‘00000 S2--S0’	No swapping/reflection. The checksum is aligned with the polynomial degree/length. The bits S3 to S31 are ’0’.
	‘1’	32	S24---S31	S16---S23	S8---S15	S0---S7	Byte aligned checksum reflection.
		21	‘0000 0000’	‘S16---S20 000’	S8---S15	S0---S7	Bit reflection. The checksum is byte aligned. Bit S21-S23 and S24-S31 are ‘0’.
		16	‘0000 0000’	‘0000 0000’	S8---S15	S0---S7	Bit reflection. The checksum is byte aligned. Bit S16-S31 are ‘0’.
		3	‘0000 0000’	‘0000 0000’	‘0000 0000’	‘S0---S2 00000’	Bit reflection. The checksum is byte aligned. Bit S3-S7 and S8-S31 are ‘0’.

Table 8-10. Preliminary checksum #1 bit-wise and/or byte-wise reflection/swapping

ROBYT	ROBIT	LEN	Preliminary checksum #2				
			+3	+2	+1	+0	Action
‘1’	‘0’	32	S7---S0	S15---S8	S23---S16	S31---S24	Byte aligned checksum swapping.
		21	‘0000 0000’	S7---S0	S15---S8	‘000 S20---S16’	Byte swapping. The checksum is byte aligned. Bit S21-S23 and S24-S31 are ‘0’.
		16	‘0000 0000’	‘0000 0000’	S7---S0	S15---S8	Byte aligned checksum swapping. Bit S16-S31 are ‘0’.
		3	‘0000 0000’	‘0000 0000’	‘0000 0000’	‘00000 S2_S0’	No byte swapping. Bit S3-S7 and S8-S31 are ‘0’.
	‘1’	32	S0---S7	S8---S15	S16---S23	S24---S31	Bit reflection and byte swapping aligned with polynomial degree/length.
		21	‘0000 0000’	‘000 S0---S4’	S5---S12	S13---S20	Bit reflection and byte swapping. The checksum is aligned with polynomial length/ degree. Bit S21-S23 and S24- S31 are ‘0’.
		16	‘0000 0000’	‘0000 0000’	S0---S7	S8---S15	Bit reflection and byte swapping. The checksum is aligned with polynomial length/ degree. Bit S16-S31 are ‘0’.
		3	‘0000 0000’	‘0000 0000’	‘0000 0000’	‘00000 S0---S2’	Bit reflection and byte swapping. The checksum is aligned with polynomial length/ degree. Bit S3-S7 and S8-S31 are ‘0’.

7. The checksum after applying settings of CRCn_CFG:ROBIT/ROBYT is 'preliminary checksum #2'.
8. The 'preliminary checksum #2' is XOR’ed with the contents of CRCn_FXOR register to get the 'final checksum'.
9. The 'final checksum' gets available at CRCn_RD register.

8.3.3 CRC calculation example

Consider the following values for calculating 8-bit CRC checksum value.

Input data = 0x0F (Hex)

Polynomial = $x^8 + x^2 + x + 1$

Seed = 0xFF (Hex) Final XOR = 0x00 (Hex)

The coefficients of the polynomial are arranged in Table 8-11.

Table 8-11. Coefficients of the polynomial

x^8	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0
1	0	0	0	0	0	1	1	1

The highest order coefficient x8 provides the degree of the CRC polynomial and the checksum length, respectively. It must not be set to '1' while configuring CRCn_POLY register, instead CRCn_CFG:LEN should be configured (here it is CRCn_CFG:LEN = 8). Therefore, the value of the polynomial that is written to CRCn_POLY register in accordance with above coefficients is 0x07 (Hex).

The input/output bit reflection is disabled in this example.

The Programmable CRC registers should be configured as follows for the given values:

The CRC configuration register is configured by considering 8-bit input data size and 8-bit polynomial/checksum length as follows.

CRCn_CFG = 0x00080000 (Hex)

CRCn_POLY = 0x00000007 (Hex)

CRCn_SEED = 0x000000FF (Hex)

CRCn_FXOR = 0x00000000 (Hex)

CRCn_WR = 0x0000000F (Hex)

The final result of CRC checksum calculation is 0xDE (Hex), which gets available after 10 clock cycles (once CRCn_WR is written) in the CRCn_RD register.

If another input data is given to the CRC module, then 'preliminary checksum #1' (0xDE) is used as the initial seed value.

If the new CRC calculation should start from the seed value instead of from the last CRC result, then the CRCn_SEED register needs to be re-written (even if it is the same seed value as before).

9. Tightly Coupled Flash

This chapter explains the function and operation of the Tightly Coupled Flash (TCFLASH).

9.1 Outline of the TCFLASH

This section describes the features and block diagram of the Tightly Coupled Flash, hereafter referred to as TCFLASH.

Features of TCFLASH

The TCFLASH memory of the system is mapped to both low latent non-cacheable Tightly Coupled Memory (TCM) address space and cacheable Advanced eXtensible Interface (AXI) address space with higher latency. Flash write or erase can be carried out using instructions from the CPU or any other master via the TCFLASH Interface (TCFLASH_IF) circuit, which also offers error correction, and protects the Flash against unprivileged and unsecured accesses. The TCFLASH can be programmed by the Flash Parallel Programmer. Features of the TCFLASH are listed below.

- It supports low latent non-cacheable access via TCM. The TCFLASH is mapped to certain address spaces, and access via AXI are cacheable but with higher latency
- Upto 8 MB of address space are allocated for TCFLASH in both the TCM and AXI
- It supports read access only via TCM, while read and write accesses are supported via AXI
- AXI write has the highest priority. TCM read and AXI read have balanced priority
- Flash read by the CPU can achieved with 8-,16-, 32- or 64-bit accesses
- Flash writing by the CPU is restricted to 32-bit access if the Error Correction Code (ECC) is enabled; 8-,16- and 32-bit accesses are allowed when the ECC is disabled
- It supports unaligned read access. Unaligned write access is not supported and leads to a slave error response
- Flash programming can be done in either direct mode (by a Flash Parallel Programmer) or CPU mode (through instructions from the CPU or from a JTAG debugger)
- Flexible programming of the Flash access wait cycles is possible to achieve the best performance at any selected system clock frequency
- The TCFLASH supports interleaved access of large sectors of even and odd Flash macros to achieve full performance - even with one wait cycle setting. Small sectors are not interleaved so as to keep more sectors available for independent use
- It supports an ECC scheme for Single-bit Error Correction and Double-bit Error Detection (SEC- DED), the same as the 32-bit ECC scheme used in the Arm Cortex-R4
- ECC checking for the TCM interface is done inside the Arm Cortex-R4
- The ECC logic can be tested by injecting Flash read data with errors. Error injection is effective for accesses via TCM and AXI
- Flash configuration register settings are 32-bit key protected to ensure safe operation
- Read, write and execute access permission per sector is based on settings provided by the Flash security module
- Flash write or erase is only possible in MCU privileged mode. Any attempt to write or erase the Flash in non-privileged mode leads to a slave error response
- Flash write or erase to a protected sector resets the Flash memory and generates a slave error response
- Read access to a protected sector outputs default data and results in a slave error response
- Accessing an unused address range outputs undefined data and leads to a slave error response
- The Flash can be programmed/erased simultaneously

Block diagram of TCFLASH

Figure 9-1. TCFLASH connectivity in the system

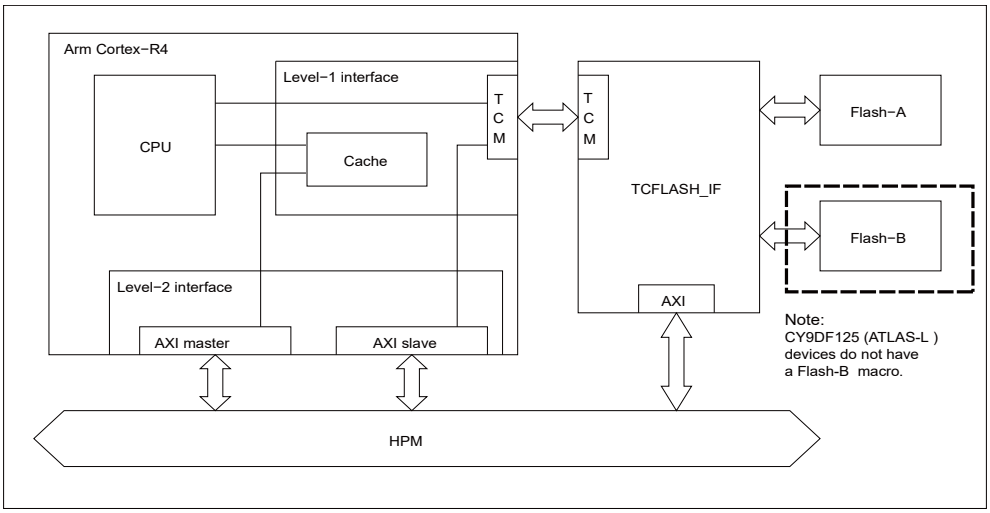
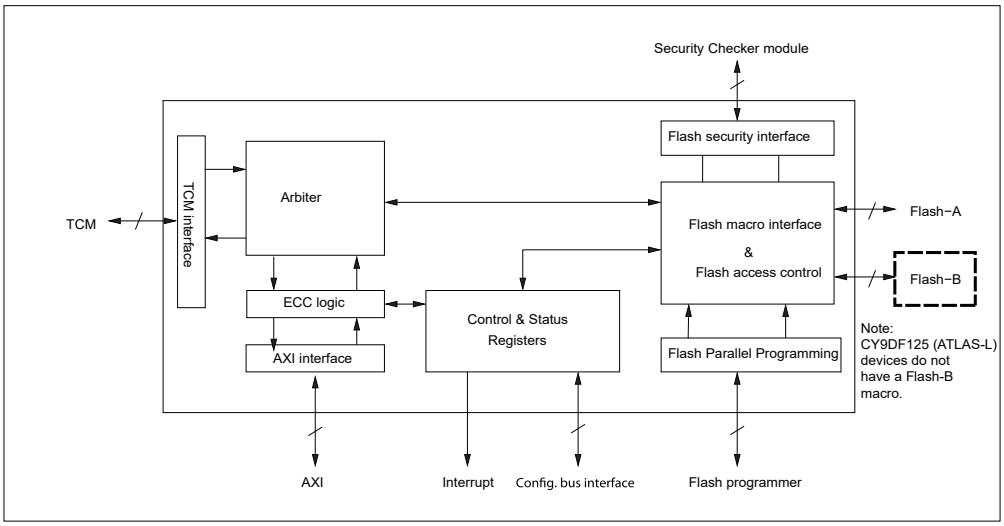


Figure 9-2. Block diagram of TCFLASH_IF



9.2 TCFLASH registers

This section describes the registers of the TCFLASH.

Registers of the TCFLASH

The following registers are available for the TCFLASH:

- Flash Configuration Protection Key Register (TCFCFG _FCPROTKEY)
- Flash Configuration Register (TCFCFG_FCFGR)
- Flash ECC Control Register (TCFCFG_FECCCTRL)
- Flash Data Bit Error Injection Register (TCFCFG_FDATEIR)
- Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR)
- Flash Interrupt Control Register (TCFCFG_FICTRLn)
- Flash Status Register (TCFCFG_FSTATn)
- Flash SEC Interrupt Register (TCFCFG_FSECIR)
- Flash ECC Error Address Register (TCFCFG_FECCEAR)
- Flash Interface Module Identification Register (TCFCFG_FMIDR)
- Flash CAM Output Lower Register (TCFCFG_FCAMLRn)
- Flash CAM Output Upper Register (TCFCFG_FCAMHRn)

Note: The suffix 'n' in the Flash Interrupt Control Register (TCFCFG_FICTRLn), the Flash Status Register (TCFCFG_FSTATn) and the Flash CAM Output Registers (TCFCFG_FCAMLRn, TCFCFG_FCAMHRn) indicates there are 'n' such registers, where 'n' is the number of Flash macros.

Memory layout of TCFLASH registers

Table 9-1. Memory layout of TCFLASH registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	reserved 00000000 00000000 00000000 00000000				TCFCFG_FCPROTKEY 00000000 00000000 00000000 00000000			
0x00000008	reserved 00000000 00000000 00000000 00000000				TCFCFG_FCFGR 00000000 00000000 00000000 00000001			
0x00000010	reserved 00000000 00000000 00000000 00000000				TCFCFG_FECCCTRL 00000000 00000000 00000000 00000000			
0x00000018	TCFCFG_FECCEIR 00000000 00000000 00000000 00000000				TCFCFG_FDATEIR 00000000 00000000 00000000 00000000			
0x00000020	TCFCFG_FICTRL1 00000000 00000000 00000000 00000000				TCFCFG_FICTRL0 00000000 00000000 00000000 00000000			
0x00000028	TCFCFG_FICTRL3 00000000 00000000 00000000 00000000				TCFCFG_FICTRL2 00000000 00000000 00000000 00000000			
0x00000030	reserved 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000038	TCFCFG_FSTAT1 00000000 00000000 00000000 00000000				TCFCFG_FSTAT0 00000000 00000000 00000000 00000000			
0x00000040	TCFCFG_FSTAT3 00000000 00000000 00000000 00000000				TCFCFG_FSTAT2 00000000 00000000 00000000 00000000			
0x00000048	reserved 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000050	TCFCFG_FECCEAR 00000000 00000000 00000000 00000000				TCFCFG_FSECIR 00000000 00000000 00000000 00000000			
0x00000058	reserved 00000000 00000000 00000000 00000000				TCFCFG_FMIDR 00000000 00000000 00000000 00000000			
0x00000060	TCFCFG_FCAMHR0 00000000 00000000 000XXXXX XXXXXXXX				TCFCFG_FCAMLR0 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000068	TCFCFG_FCAMHR1 00000000 00000000 000XXXXX XXXXXXXX				TCFCFG_FCAMLR1 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000070	TCFCFG_FCAMHR2 00000000 00000000 000XXXXX XXXXXXXX				TCFCFG_FCAMLR2 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000078	TCFCFG_FCAMHR3 00000000 00000000 000XXXXX XXXXXXXX				TCFCFG_FCAMLR3 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			

Note: The register map is given for the TCFlash configuration of the FCR4 Cluster series (i.e. two flash macros). Addresses 0x28 to 0x34, 0x40 to 0x4C, and 0x70 to 0x8C are reserved for multiple instances of the TCFCFG_FICTRLn, TCFCFG_FSTATn, TCFCFG_FCAMLRn and TCFCFG_FCAMHRn registers, one for each Flash macro in the system.

9.2.1 Flash Configuration Protection Key Register (TCFCFG _FCPROTKEY)

The Flash Configuration Protection Key Register (TCFCFG_FCPROTKEY) should be written with the correct key (0xCF61F1A5) before each write access to the Flash Configuration Register (TCFCFG_FCFGR), the Flash ECC Control Register (TCFCFG_FECCCTRL) or the Flash ECC Error Injection Registers (TCFCFG_FDATEIR and TCFCFG_FECCEIR). Access to these registers is locked again after the next write on the Advanced High-Performance Bus (AHB) configuration bus. An incorrect key, writing to the Flash Configuration registers without unlocking or any deviation from the above protection sequence generates an error response on the slave bus and hence a data abort. This register should be written to using only 32-bit writes.

Flash Configuration Protection Key Register (TCFCFG_FCPROTKEY)

Figure 9-3. Flash Configuration Protection Register (TCFCFG_FCPROTKEY)

TCFCFG_FCPROTKEY																															
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**
FCPROTKEY[31]	FCPROTKEY[30]	FCPROTKEY[29]	FCPROTKEY[28]	FCPROTKEY[27]	FCPROTKEY[26]	FCPROTKEY[25]	FCPROTKEY[24]	FCPROTKEY[23]	FCPROTKEY[22]	FCPROTKEY[21]	FCPROTKEY[20]	FCPROTKEY[19]	FCPROTKEY[18]	FCPROTKEY[17]	FCPROTKEY[16]	FCPROTKEY[15]	FCPROTKEY[14]	FCPROTKEY[13]	FCPROTKEY[12]	FCPROTKEY[11]	FCPROTKEY[10]	FCPROTKEY[9]	FCPROTKEY[8]	FCPROTKEY[7]	FCPROTKEY[6]	FCPROTKEY[5]	FCPROTKEY[4]	FCPROTKEY[3]	FCPROTKEY[2]	FCPROTKEY[1]	FCPROTKEY[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-2. Flash Configuration Protection Register (TCFCFG_FCPROTKEY) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FCPROTKEY	Flash Configuration Protection Key This register protects the Flash Configuration Register (TCFCFG_FCFGR), the Flash ECC Control Register (TCFCFG_FECCCTRL) and the Flash ECC Error Injection Registers (TCFCFG_FDATEIR and TCFCFG_FECCEIR) from being accidentally modified by software. Writing the correct key value (0xCF61F1A5) to this register unlocks write access to the TCFCFG_FCFGR, TCFCFG_FECCCTRL, TCFCFG_FDATEIR and TCFCFG_FECCEIR registers. All other values will cause a data abort. Reading this register always returns 0xFFFFFFFF when write access to TCFCFG_FCFGR, TCFCFG_FECCCTRL, TCFCFG_FDATEIR and TCFCFG_FECCEIR is unlocked. If, on the other hand, write access is locked (default), then the value returned when reading this register is always 0x00000000.

9.2.2 Flash Configuration Register (TCFCFG_FCFGR)

The Flash Configuration Register (TCFCFG_FCFGR) controls the wait cycle settings for Flash access. It also contains the enable/disable bit for Flash write access, the Flash reset control and status bits, and it determines TCM and AXI priority. Write access to this register is only possible if the correct key is written to the TCFCFG_FCPROTKEY register.

Flash Configuration Register (TCFCFG_FCFGR)

Figure 9-4. Flash Configuration Register (TCFCFG_FCFGR)

TCFCFG_FCFGR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SWFRST	TCMPR	WE	read0	read0	FAWC[1]	FAWC[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WPS1	RWPS	RWPS	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 9-3. Flash Configuration Register (TCFCFG_FCFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	SWFRST	Software Triggered Flash Reset This bit is used to reset the Flash macro by software. '0': No effect '1': Triggers Flash macro reset. As soon as the Flash Ready Status (TCFCFG_FSTATn:RDY) flag is read as '1', the Flash macro is ready to read or write the next command Reading this bit returns '0'.
[5]	TCMPR	TCM Priority Enable This bit enables/disables Tightly Coupled Memory (TCM) priority. '0': TCM and Advanced eXtensible Interface (AXI) accesses follow balanced priority (default) '1': TCM access has higher priority than AXI access **Note:** For more details on TCM and AXI priority refer to 9.8 Notes on using Flash memory.

Table 9-3. Flash Configuration Register (TCFCFG_FCFGR) bits

Bit Position	Bit Field Name	Bit Description
[4]	WE	Flash Write Enable This bit enables/disables writing to the Flash. '0': Flash write is disabled (default) '1': Flash write is enabled Write access to the Flash with this bit set to '0' results in a bus error. Setting this bit to '1' disables prefetch access to the interleaved address and thus reduces read performance. --This is to avoid hardware sequence flags (of the Flash program) and because prefetch operations disturb each other.--
[3:2]	read0	-
[1:0]	FAWC	Flash Access Wait Cycle Control These bits define the number of wait cycles required for Flash read and write accesses. They should be set based on the system clock frequency and Flash access time. '00': 0 wait cycles '01': 1 wait cycle (default) '10': 2 wait cycles '11': 3 wait cycles The relationship between the wait cycle setting and clock frequency can be given as: FAWC = (System clock frequency/Flash operating frequency) - 1 For example, for a system clock frequency = 160 MHz and a Flash operating frequency = 80 MHz, then FAWC = '01'.

9.2.3 Flash ECC Control Register (TCFCFG_FECCCTRL)

The Flash ECC Control Register (TCFCFG_FECCCTRL) controls the disabling of the ECC logic. This register is accessible only if the correct key is written to the TCFCFG_FCPROTKEY register. TCFCFG_FECCCTRL is writable to only once; any further writes will lead to a bus error response.

Flash ECC Control Register (TCFCFG_FECCCTRL)

Figure 9-5. Flash ECC Control Register (TCFCFG_FECCCTRL)

TCFCFG_FECCCTRL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ECCOFF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-4. Flash ECC Control Register (TCFCFG_FECCCTRL)

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ECCOFF	ECC Disable/Switch-Off This bit disables Error Corrrection Code (ECC) generation and the checking of AXI accesses. '0': ECC generation/checking is ON (default) '1': ECC generation/checking is OFF **Note:** ECC generation and the checking for TCM access is enabled or disabled inside the Arm® Cortex®-R4.

9.2.4 Flash Data Bit Error Injection Register (TCFCFG_FDATEIR)

The Flash Data Bit Error Injection Register (TCFCFG_FDATEIR) injects errors into Flash read data to test the ECC logic. Error injection is carried out on both the upper and lower 32 bits of the Flash read data at the same time. This register is writable to only if it has been unlocked by writing the correct key to the TCFCFG_FCPROTKEY register.

Flash Data Bit Error Injection Register (TCFCFG_FDATEIR)

Figure 9-6. Flash Data Bit Error Injection Register (TCFCFG_FDATEIR)

TCFCFG_FDATEIR																															
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**
FDATEIR[31]	FDATEIR[30]	FDATEIR[29]	FDATEIR[28]	FDATEIR[27]	FDATEIR[26]	FDATEIR[25]	FDATEIR[24]	FDATEIR[23]	FDATEIR[22]	FDATEIR[21]	FDATEIR[20]	FDATEIR[19]	FDATEIR[18]	FDATEIR[17]	FDATEIR[16]	FDATEIR[15]	FDATEIR[14]	FDATEIR[13]	FDATEIR[12]	FDATEIR[11]	FDATEIR[10]	FDATEIR[9]	FDATEIR[8]	FDATEIR[7]	FDATEIR[6]	FDATEIR[5]	FDATEIR[4]	FDATEIR[3]	FDATEIR[2]	FDATEIR[1]	FDATEIR[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-5. Flash Data Bit Error Injection Register (TCFCFG_FDATEIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FDATEIR	Flash Data bit Error Injection This register is used to inject errors into Flash read data bits to test ECC logic. Error injection is done by inverting certain Flash read data bits before passing the data to the ECC checking logic. '0': Has no effect on the corresponding bit of the Flash read data (default) '1': Flips the corresponding bit of the Flash read data

9.2.5 Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR)

The Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR) is used to inject errors into the ECC bits of Flash read data and to test the ECC logic. Error injection affects both the upper and lower 7 bits of the Flash ECC output at the same time. This register is writable to only if it has been unlocked by writing the correct key the to TCFCFG_FCPROTKEY register.

Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR)

Figure 9-7. Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR)

TCFCFG_FECCEIR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	FECCEIR[6]	FECCEIR[5]	FECCEIR[4]	FECCEIR[3]	FECCEIR[2]	FECCEIR[1]	FECCEIR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-6. Flash ECC Bit Error Injection Register (TCFCFG_FECCEIR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6:0]	FECCEIR	Flash ECC bit Error Injection This register is used to inject errors in ECC bits of Flash read data to test ECC logic. Error injection is done by inverting respective ECC bits of Flash read data before passing it to ECC checking logic. '0': Has no effect on corresponding ECC bit of the Flash read data (default) '1': Flips corresponding ECC bit of the Flash read data

9.2.6 Flash Interrupt Control Register (TCFCFG_FICTRLn)

The Flash Interrupt Control Register (TCFCFG_FICTRLn) contains the interrupt enable and clear bits of the TCFCFG_FSTATn:RDY and TCFCFG_FSTATn:HANG interrupts. It also contains an ECC write flag clear bit. Each physical Flash macro has its own TCFCFG_FICTRL register.

Flash Interrupt Control Register (TCFCFG_FICTRLn)

Figure 9-8. Flash Interrupt Control Register (TCFCFG_FICTRLn)

TCFCFG_FICTRL0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WR32FC	HANGIC	RDYIC	read0	read0	read0	read0	read0	read0	HANGIE	RDYIE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0WP1	R0WP1	R0	R0	R0	R0	R0	R0	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-7. Flash Interrupt Control Register (TCFCFG_FICTRLn) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10]	WR32FC	32-bit Write Flag Clear '0': No effect '1': Clears the 32-bit Write Control Flag (TCFCFG_FSTATn:WR32F) Reading this bit returns '0'.
[9]	HANGIC	Hangup-1 Interrupt Clear '0': No effect '1': Clears the Hangup-1 Interrupt status bit(TCFCFG_FSTATn:HANGINT) Reading this bit returns '0'.
[8]	RDYIC	Flash Ready Interrupt Clear '0': No effect '1': Clears the Flash Ready Interrupt status bit (TCFCFG_FSTATn:RDYINT) Reading this bit returns '0'.
[7:2]	read0	-
[1]	HANGIE	Hangup-1 Interrupt Enable '0': Disables hangup-1 interrupt (default) '1': Enables hangup-1 interrupt

Table 9-7. Flash Interrupt Control Register (TCFCFG_FICTRLn) bits

Bit Position	Bit Field Name	Bit Description
[0]	RDYIE	Flash Ready Interrupt Enable '0': Disables Flash ready interrupt (default) '1': Enables Flash ready interrupt

9.2.7 Flash Status Register (TCFCFG_FSTATn)

The Flash Status Register (TCFCFG_FSTATn) holds the status of TCFCFG_FSTATn:RDY and TCFCFG_FSTATn:HANG outputs of the Flash memory and their respective interrupt status. It also stores the 32-bit write control flag. Each physical Flash macro has a TCFCFG_FSTAT register. As a read-only register; any attempt to write to it will result in a bus error response.

Flash Status Register (TCFCFG_FSTATn)

Figure 9-9. Flash Status Register (TCFCFG_FSTATn)

TCFCFG_FSTAT0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	HANGINT	RDYINT	read0	read0	read0	WR32F	read0	read0	HANG	RDY
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-8. Flash Status Register (TCFCFG_FSTATn) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	HANGINT	Hangup-1 Interrupt This bit is set on the rising edge of the HANG output. It is cleared by writing '1' to the Hangup-1 Interrupt Clear bit (TCFCFG_FICTRLn:HANGIC). '0': Hangup-1 condition has not occurred (default) '1': Hangup-1 condition has occurred
[8]	RDYINT	Flash Ready Interrupt This bit is set on the rising edge of the RDY output. It is cleared by writing '1' to the RDY interrupt clear bit (TCFCFG_FICTRLn:RDYIC). '0': Rising edge of Flash RDY output has not been detected (default) '1': Rising edge of Flash RDY output has been detected (i.e. Flash write or erase operation is completed)
[7:5]	read0	-

Table 9-8. Flash Status Register (TCFCFG_FSTATn) bits

Bit Position	Bit Field Name	Bit Description
[4]	WR32F	32-bit Write Control Flag This bit is valid for 32-bit write accesses only. It is toggled every time a Flash write sequence is completed. The 32-bit Write Control Flag can be cleared by writing '1' to the 32-bit write flag clear bit (TCFCFG_FICTRLn:WR32FC). It is also cleared if the TCFCFG_FCFGR:WE bit is '0'. '0': The Flash interface writes the lower half of the word to Flash (default) '1': The Flash interface writes the upper half of the word to Flash with ECC. The ECC check is temporarily disabled when this bit is '1'
[3:2]	read0	-
[1]	HANG	Hangup-1 Status This bit indicates whether or not the Flash macro is in Hangup-1 state. '0': Flash macro is in normal state (default) '1': Flash macro is in Hangup-1 state This bit is set to '1' when: a. '1' is written to a memory cell containing '0' b. The automatic algorithm is not completed within a restricted time frame **Note:** When Flash macro enters Hangup-1 state, it can be brought back to normal state by either a hard reset, a reset command or by a software triggered Flash reset (i.e. TCFCFG_FCFGR:SWFRST).
[0]	RDY	Flash Ready Status This bit reflects the status (sampled) of the Flash RDY output. It indicates whether or not the Flash macro is ready to accept a new command. '0': Flash auto-algorithm (write or erase) is in progress - only a read or suspend command is accepted '1': Flash is returned from auto-algorithm (write or erase) and is ready for a new command (default) The RDY bit is pulled low during Flash reset and is set to '1' after reset has completed.

9.2.8 Flash SEC Interrupt Register (TCFCFG_FSECIR)

The Flash SEC Interrupt Register (TCFCFG_FSECIR) contains the status of the ECC single bit error interrupt as well as the corresponding interrupt enable and clear bits.

Flash SEC Interrupt Register (TCFCFG_FSECIR)

Figure 9-10. Flash SEC Interrupt Register (TCFCFG_FSECIR)

TCFCFG_FSECIR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SECINT	read0	read0	read0	read0	read0	read0	read0	SECIC	read0	read0	read0	read0	read0	read0	read0	SECIE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-9. Flash SEC Interrupt Register (TCFCFG_FSECIR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	SECINT	ECC Single Error Correction Interrupt This read-only bit is set when a single error is detected or corrected during ECC checking. It is cleared by writing '1' to the ECC Single Error Correction Interrupt Clear bit (TCFCFG_FSECIR:SECIC). '0': An ECC single error has not occurred (default) '1': An ECC single error has occurred
[15:9]	read0	-
[8]	SECIC	ECC Single Error Correction Interrupt Clear '0': No effect '1': Clears the ECC Single Error Correction Interrupt bit (TCFCFG_FSECIR:SECINT) Reading this bit returns '0'.
[7:1]	read0	-
[0]	SECIE	ECC Single Error Correction Interrupt Enable '0': ECC Single Error Correction Interrupt disabled (default) '1': ECC Single Error Correction Interrupt enabled

9.2.9 Flash ECC Error Address Register (TCFCFG_FECCEAR)

The Flash ECC Error Address Register (TCFCFG_FECCEAR) stores the address of the Flash memory location where the ECC single bit error occured. Software can use this register to mark the error location. ECC checking of the upper and lower 32 bits of each line of Flash read data is always carried out in parallel and the 64-bit aligned address is stored in this register, independent of the error location. This is a read-only register; any attempt to write to it will lead to a bus error response.

Flash ECC Error Address Register (TCFCFG_FECCEAR)

Figure 9-11. Flash ECC Error Address Register (TCFCFG_FECCEAR)

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

Table 9-10. Flash ECC Error Address Register (TCFCFG_FECCEAR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FECCEAR	Flash ECC Error Address Register This read-only register contains the Flash address of where the latest ECC single bit error has occurred. This address may be helpful in marking the memory location as a weak cell. **Note:** Only AXI accesses are logged. For TCM, the Arm® Cortex®-R4 contains the Correctable Error Location Register.

9.2.10 Flash Interface Module Identification Register (TCFCFG_FMIDR)

The Flash Interface Module Identification Register (TCFCFG_FMIDR) contains the identification number, revision and patch details of the TCFLASH module. This is read- only register; any attempt to write to it will lead to a bus error response

Flash Interface Module Identification Register (TCFCFG_FMIDR)

Figure 9-12. Flash Interface Module Identification Register (TCFCFG_FMIDR)

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

Table 9-11. Flash Interface Module Identification Register (TCFCFG_FMIDR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	TCFLASH_IF Module ID The TCFLASH_IF module implemented in the device may vary from device to device. This register identifies the hardware version used in the device. **Note:** For more details on the version number, refer to the device specific datasheet.

9.2.11 Flash CAM Output Lower Register (TCFCFG_FCAMLRn)

The Flash CAM Output Lower Register (TCFCFG_FCAMLRn) stores the lower 32 bits [31:0] of the TCFCFG_FCAMLRn:CAM output (i.e. the identification number) of a particular Flash macro. Each physical Flash macro has a TCFCFG_FCAML register. This is a read-only register; any attemp to write to it will lead to bus error response.

Flash CAM Output Lower Register (TCFCFG_FCAMLRn)

Figure 9-13. Flash CAM Output Lower Register (TCFCFG_FCAMLRn)

TCFCFG_FCAMLR0																															
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**
CAML[31]	CAML[30]	CAML[29]	CAML[28]	CAML[27]	CAML[26]	CAML[25]	CAML[24]	CAML[23]	CAML[22]	CAML[21]	CAML[20]	CAML[19]	CAML[18]	CAML[17]	CAML[16]	CAML[15]	CAML[14]	CAML[13]	CAML[12]	CAML[11]	CAML[10]	CAML[9]	CAML[8]	CAML[7]	CAML[6]	CAML[5]	CAML[4]	CAML[3]	CAML[2]	CAML[1]	CAML[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 9-12. Flash CAM Output Lower Register (TCFCFG_FCAMLRn) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CAML	Flash CAM Output Lower Word This is a read-only register. Reading this register returns the lower part of the Flash macro CAM output. The CAM output gives the unique identification number of the Flash macro.

9.2.12 Flash CAM Output Upper Register (TCFCFG_FCAMHRn)

Flash CAM Output Upper Register (TCFCFG_FCAMHRn) stores the upper 13 bits [44:32] of TCFCFG_FCAMLRn:CAM output (i.e. the identification number) of the Flash macro. Each physical Flash macro has a TCFCFG_FCAMH register. This is a read-only register; any attempt to write to this register will lead to bus error response.

Flash CAM Output Upper Register (TCFCFG_FCAMHRn)

Figure 9-14. Flash CAM Output Upper Register (TCFCFG_FCAMHRn)

TCFCFG_FCAMHR0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CAMH[12]	CAMH[11]	CAMH[10]	CAMH[9]	CAMH[8]	CAMH[7]	CAMH[6]	CAMH[5]	CAMH[4]	CAMH[3]	CAMH[2]	CAMH[1]	CAMH[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 9-13. Flash CAM Output Upper Register (TCFCFG_FCAMHRn) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:13]	read0	-
[12:0]	CAMH	Flash CAM Output Higher Word This is a read-only register. Reading this register returns the upper part of Flash macro CAM output. The CAM output gives the unique identification number of the Flash macro.

9.3 Operation of the TCFLASH

This section describes the operation of the TCFLASH.

Flash memory

TCFLASH is the standard Flash macro, which is optimized for high read performance. Up to two modules of this type (Flash-A and Flash-B) are available in the current MCU (This is true for CY9DF126 devices, i.e. ATLAS. CY9DF125 devices, i.e. ATLAS-L feature only one TCFLASH module). Each Flash macro has up to 8 small sectors of 8 KB size and up to 16 large sectors of 64 KB size. The features supported by a Flash macro are:

- It uses an automatic program algorithm for write and erase
- Sector erase function (any combination of sectors)
- Erase pause or resume functions
- It indicates write and macro erase command sequence detection
- Detection of completion of writing or erasing using data polling or toggle bit functions, or by CPU interrupts
- Detection of Hangup-1 state
- Flash read access time: minimum 1 machine cycle

Flash memory operation modes

TCFLASH can be operated in two different operation modes:

- CPU mode: This is the default mode of Flash operation in which the Flash can be accessed by the CPU or any other master in the system via the TCM or the AXI interface. Flash programming is allowed only through the AXI interface. The writing to or erasing of Flash is performed by programming access sequences that trigger its auto algorithm for write, erase etc. The list of command sequences supported is given in Table 9-2. ECC calculation is handled inside the Flash interface
- Flash Parallel Programming (FPP) mode: This mode is enabled by setting the mode pins MD[5:0] to 'xxx111' in board test mode. This then allows the device to behave as a standalone Flash. For direct access, the Flash interface pins are directly mapped to the external MCU pins, and write or erase is performed using a Flash memory programmer. In FPP mode the ECC bits and data bits in Flash are treated equally. Calculation of the ECC is handled by the Flash programming tool and not by the Flash interface

Flash memory address/sector mapping

Address mapping of the Flash sectors depends on the following modes of operation:

- In CPU mode, Flash memory is mapped to the TCM address space 1 for low latent non- cacheable access and to the AXI address space for cacheable higher latency accesses. TCM and AXI address mapping for devices with two TCFLASH modules are shown in Figure 9-15 and Figure 9-16, while the address mapping for devices with one module (i.e. CY9DF125) is shown in Figure 9-17 and Figure 9-18
- Address mapping of the Flash memory is such that the uppermost smaller (8 KB) sector and bottommost larger (64 KB) sector of Flash-A (i.e. SA7 and SA8) are kept at fixed positions independent of the memory size. The smaller 8K sectors grow in a downward direction while the larger 64K sectors grow in an upward direction
- For devices with two TCFLASH modules, the addressing of the big sectors of Flash-A (even) and Flash-B (odd) are interleaved (refer to Figure 9-15). This helps in accessing even and odd word addresses on alternative clocks, thereby hiding Flash access latency in case of burst or sequential access.
- In the Flash Parallel Programming mode[2], no address mapping is done. All address pins are directly connected to the Flash. Thus Flash addressing is just the physical Flash addressing as shown in Figure 9-19 with some higher address bits used for Flash macro selection

Note 1:

In Figure 9-16 only the lower 23 bits (8 MB address space) of the TCM address are shown. The base address can be adjusted using the ATCM Region Register of Arm Cortex-R4.

Note 2:

To reduce the pin count requirement, Flash Parallel Programming mode only supports 16-bit directional pins for data and 7-bit directional pins for ECC for Flash reading and writing.

Flash interleaved access and performance benefits

In devices with two TCFLASH modules, the addressing of the big sectors of Flash-A and Flash-B are interleaved in order to improve system performance despite the Flash access latency. Interleaving provides a system whereby all the even addresses (with respect to the third bit of the address) are in Flash-A and the odd addresses in Flash-B. This allows even and odd word addresses to be accessed on alternate clocks without any wait cycles in between, achieving gapless sequential/burst accesses even with a wait cycle setting of '01'. For example (using 64-bit read accesses):

1. Access to address 0x01000010 -> in Flash-A
2. Access to address 0x01000018 -> in Flash-B
3. Access to address 0x01000020 -> in Flash-A
4. Access to address 0x01000028 -> in Flash-B

Flash-A is accessed at half the rate of the bus cycles and Flash-B is accessed at half the rate of the bus transfer speed. Therefore, for a burst transfer it is possible to deliver twice the data rate with respect to the Flash access cycle time.

Small sectors of Flash are not interleaved because having more small sectors is considered more important than the code execution performance of these sectors.

TCM and AXI arbitration

AXI and TCM access to the Flash follows a priority scheme as described below:

- AXI write access has higher priority if the TCFCFG_FCFGR:WE bit is set
- AXI and TCM reads have balanced priority in which the AXI is given priority for the first 16 read accesses and TCM is given priority for the next 16 read accesses
- AXI and TCM balanced priority can be overridden by setting the TCFCFG_FCFGR:TCMPR bit to '1'. This then gives higher priority to a TCM access
- TCM and AXI access to one or the other Flash macro has no defined priority
- An ongoing burst access is not interrupted under any circumstances

Figure 9-15. Interleaving of Flash-A and Flash-B

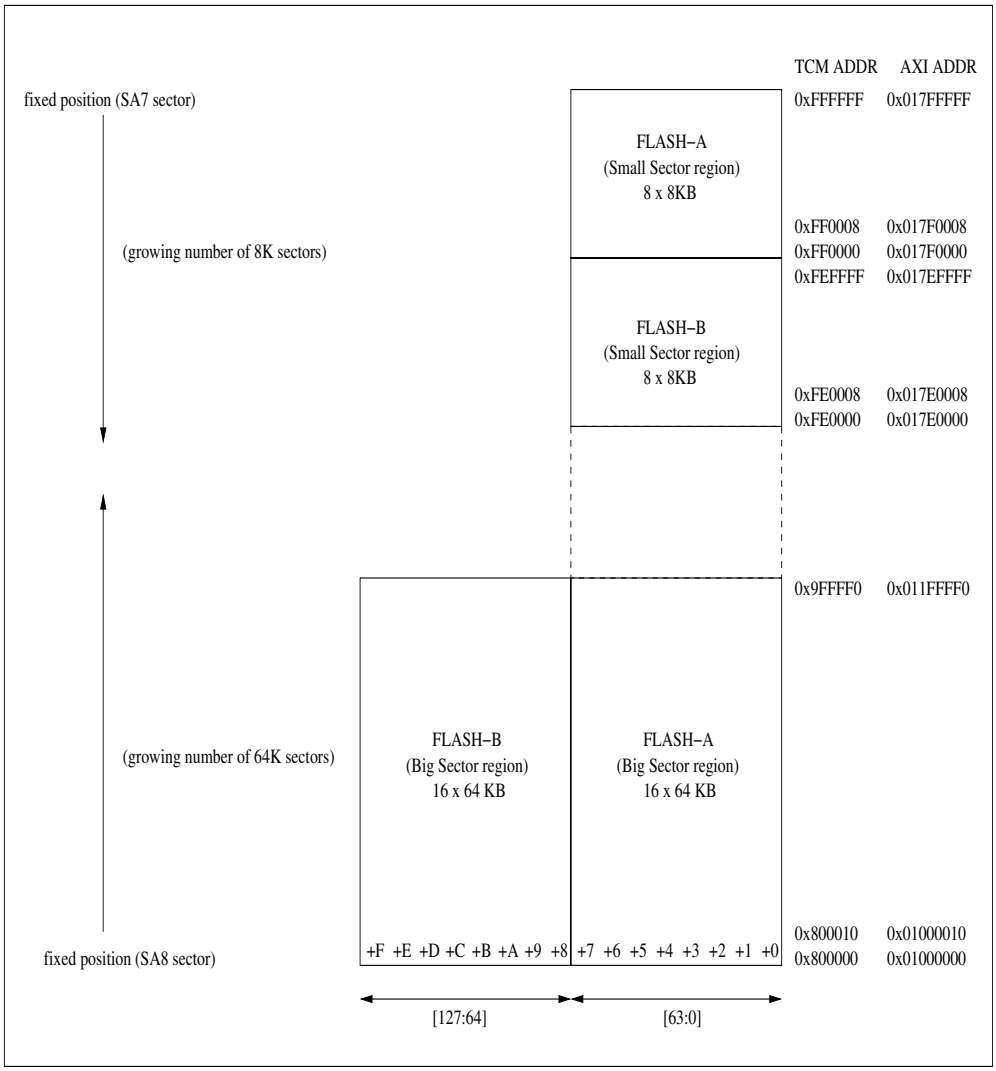
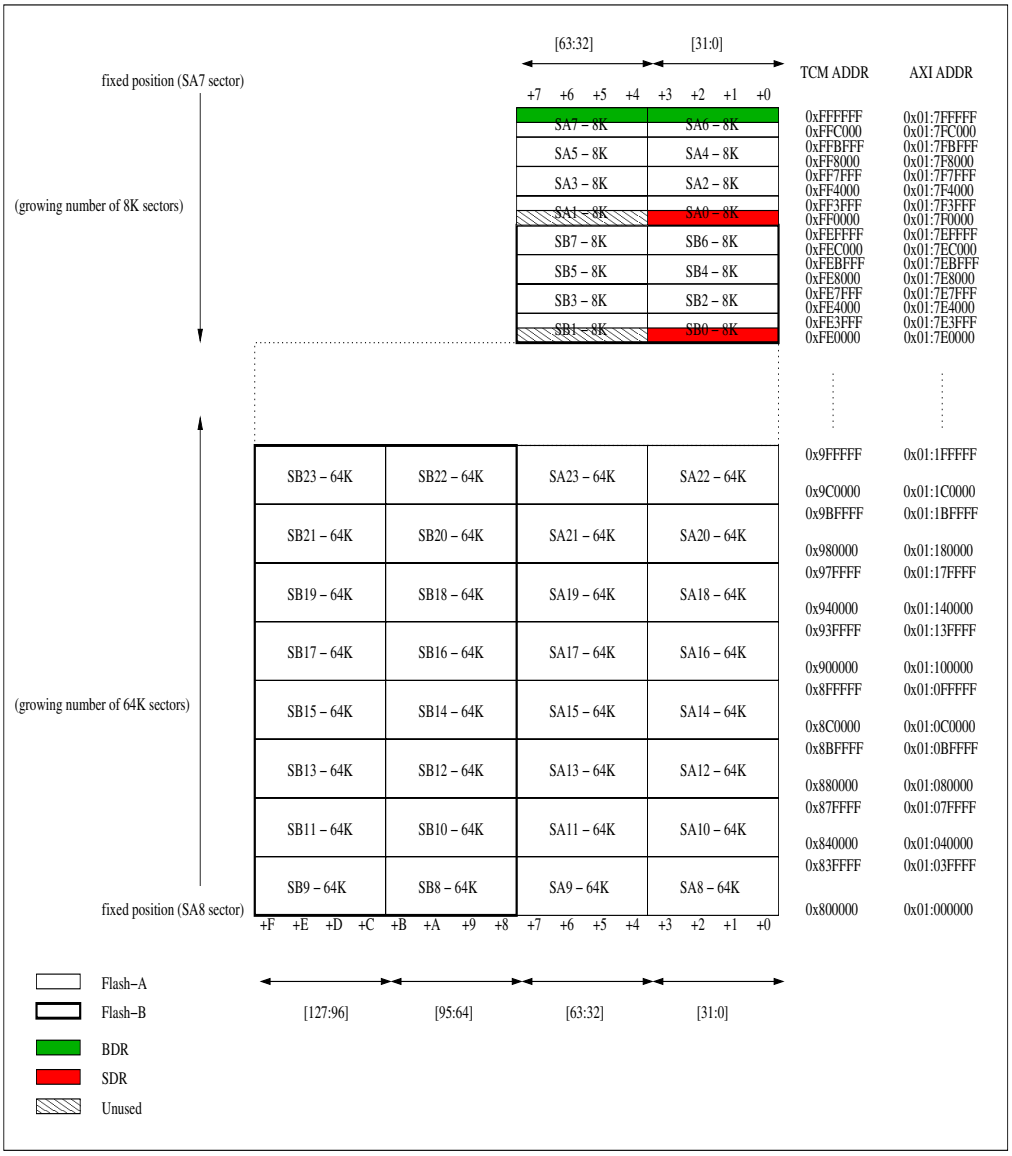


Figure 9-16. TCFLASH sector/address mapping - CPU mode



Notes:

- The Boot Description Record (BDR) is located in the uppermost addresses of sectors SA6 and SA7
- Security Description Records (SDRs) are located at the bottommost addresses of sectors SA0 and SB0

Figure 9-17. Single TCFLASH module device (no interleaving)

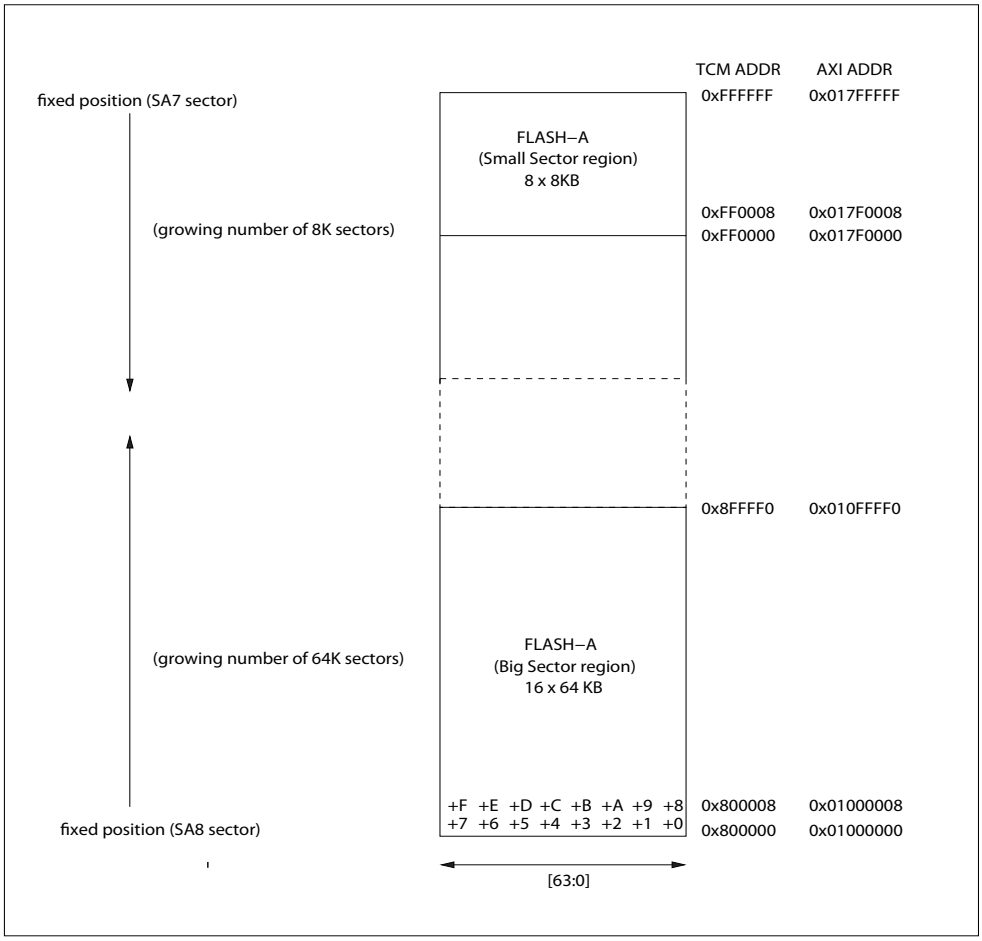
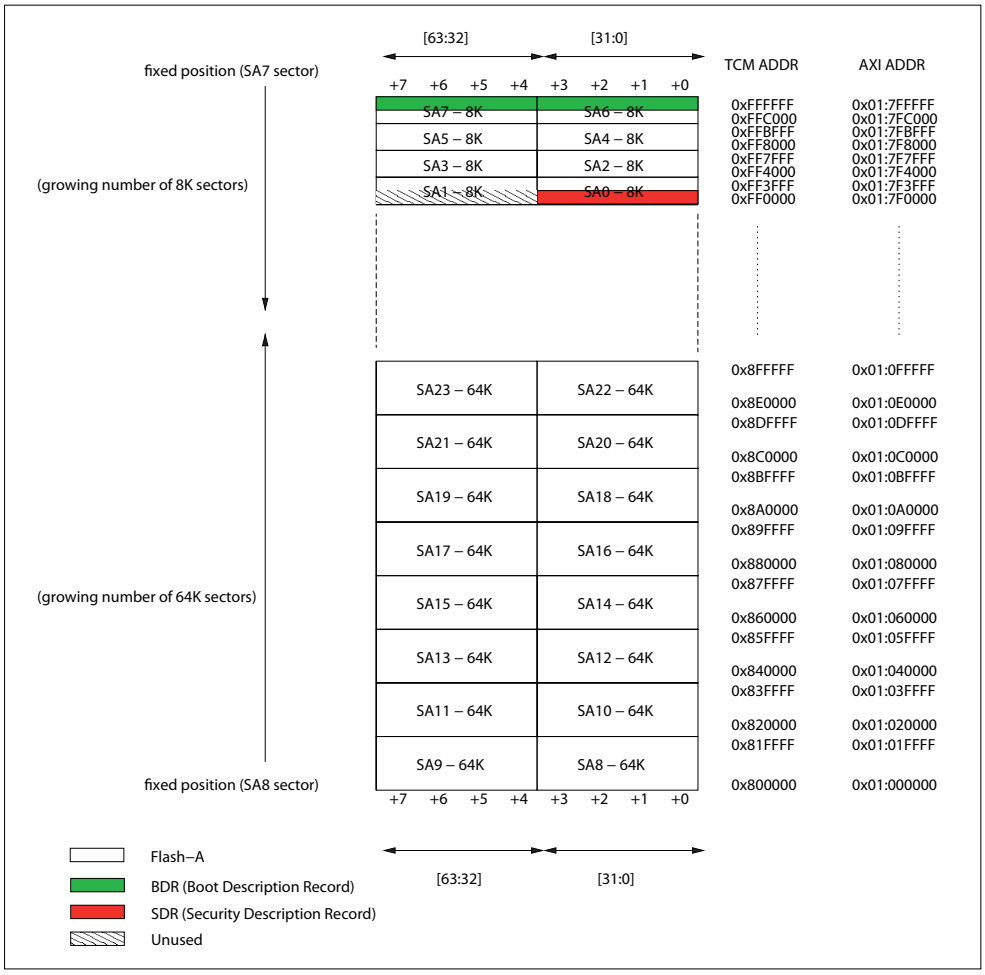


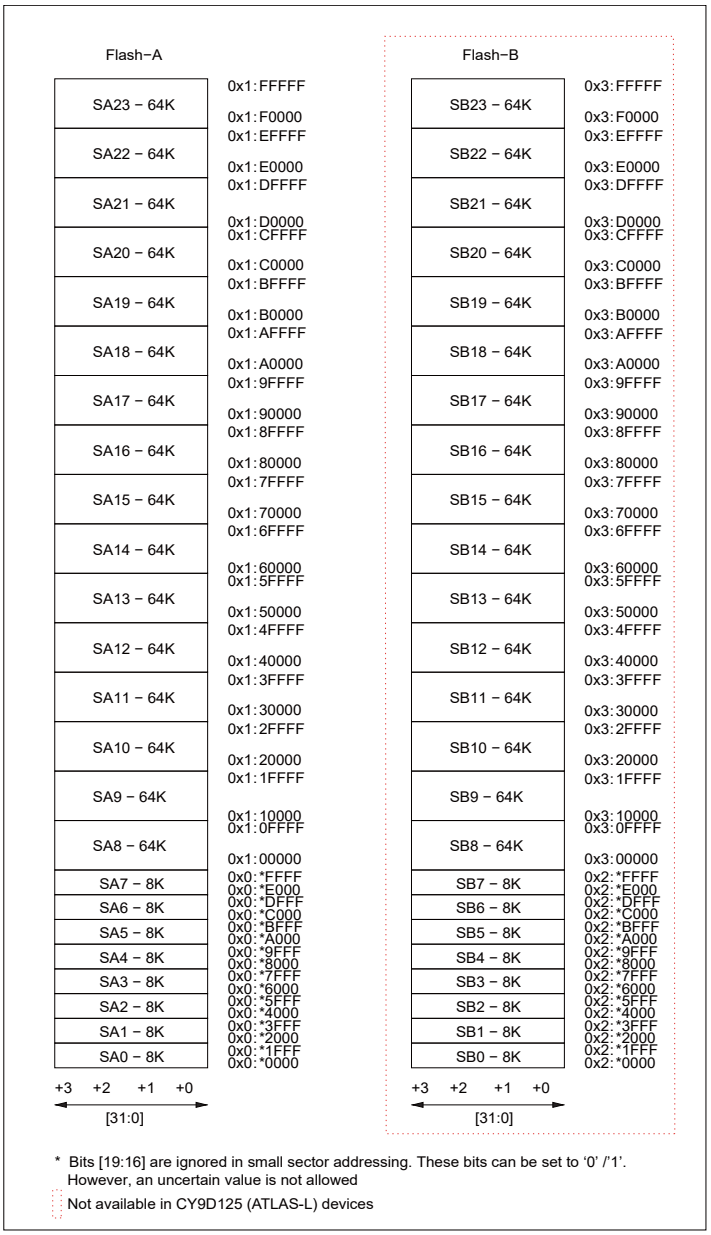
Figure 9-18. TCM and AXI address mapping for a single TCFLASH module device - CPU mode



Notes:

- The Boot Description Record (BDR) is located in the uppermost addresses of sectors SA6 and SA7
- Security Description Records (SDRs) are located at the bottommost addresses of sector SA0

Figure 9-19. TCFLASH sector or address mapping - Flash Parallel Programming (FPP) mode



Note: Higher order address bits are used for Flash selection. Address bit [21] selects between Flash-A or Flash-B.

TCFLASH programming in CPU mode

TCFLASH write or erase by the CPU or any other master follows the command sequence table given in Table 9-14. In order to handle ECC calculation and Flash writes, Flash write in CPU mode is restricted to 32- bit mode[1] and follows a defined sequence:

1. Set the Flash Write Enable bit (i.e. TCFCFG_FCFGR:WE = '1')
2. Send the Flash write command sequence (given in Table 9-14) with the Program Address (PA) and the 32-bit data to be written. 32-bit data is required to calculate the ECC from the complete data
3. With TCFCFG_FSTATn:WR32F = '0', the lower 16-bit data is written to the PA
4. Repeat the same write command sequence[2]

5. At the end of each write sequence, the TCFCFG_FSTATn:WR32F[3] bit is automatically toggled
6. With TCFCFG_FSTATn:WR32F = '1', the upper 16-bit data and ECC is written to PA + 2
7. Clear the Flash write enable bit (i.e. TCFCFG_FCFGR:WE = '0')[4]

Note 1:

If ECC is enabled, Flash write in CPU mode is restricted to 32-bit mode to allow ECC calculation. If ECC is disabled, Flash write can be 8-, 16- or 32-bits wide. There is no true 32-bit write support by the Flash memory, but the interface accepts this data type through the above-defined Flash write sequence.

Note 2:

Flash write in CPU mode should repeat the write sequence; the data for the second write must be the same as the first to ensure that the correct ECC bits have been generated and the complete 32-bit data and ECC are written correctly.

Note 3:

In the case of a 32-bit write, the ECC check is disabled after the first write sequence until the second write sequence has completed (i.e. when TCFCFG_FSTATn:WR32F = '1') because ECC data will not be consistent until the 32-bit data and ECC are completely written.

Note 4:

For safety reasons, the TCFCFG_FSTATn:WR32F flag is cleared if Flash write is disabled (i.e. TCFCFG_FCFGR:WE = '0') to avoid the temporary disabling of the ECC.

For more details concerning the TCFCFG_FCFGR:WE and TCFCFG_FSTATn:WR32F bits refer to 9.2 TCFLASH registers.

TCFLASH access wait cycles

Based on the system clock frequency and Flash access time (refer to the device specific datasheet for this value), the number of wait cycles required for Flash access can be configured by setting the TCFCFG_FCFGR:FAWC[1:0] bits (See Table 9-3 -). By default these bits are configured for 1 wait cycle. It is possible to select zero wait states for a lower system clock frequency. The same wait cycle settings are applicable for both read and write accesses to the Flash.

The configured number of wait states remain vaild until a new value is written to the configuration register (TCFCFG_FCFGR).

ECC logic

- The ECC module uses the Arm Cortex-R4's 32-bit ECC scheme for ECC encoding during Flash write and ECC syndrome decoding during Flash read
- During a write operation, the ECC module computes the ECC on 32-bit write data in the second write sequence. The upper 16-bit data along with the ECC is written to the Flash
- During a read operation, ECC syndrome is computed for the Flash read data in the same way as the corresponding check bit except that the syndrome calculation includes the check bit as well as the appropriate data bits. The syndrome calculation is therefore interpreted for no error, a single error, and double and multiple errors. Multi bit errors (i.e. more than two bit errors) can not be reliably detected
- Handling the ECC check for erased data (i.e. all data bits and ECC bits are '1', which causes an ECC error) depends on ECC syndrome used in the Arm Cortex-R4. In general, however, some of the ECC bits are flipped on both read and write paths. Erased data causes the correct ECC value; no error is generated
- The ECC module can also inject errors into the data and ECC bits read from the Flash to test ECC functionality
- The ECC can be disabled or switched off by setting a configuration register bit (TCFCFG_FECCCTRL:ECCOFF)
- Checking the ECC for TCM access is carried out inside the Arm Cortex-R4. Disabling the ECC for TCM access should be done separately in the Arm Cortex-R4 configuration register
- ECC logic is only effective in CPU mode. In FPP mode, the ECC bits are passed to the device I/O ports without any negation. Thus all data/ ECC bits of an erased Flash appear as '1'

Interrupts

The Flash memory interface generates interrupts under the following conditions:

- When both the Ready Interrupt flag (TCFCFG_FSTATn:RDYINT) and the corresponding Interrupt Enable bit (i.e. TCFCFG_FICTRLn:RDYIE) are set. TCFCFG_FSTATn:RDYINT is set when the TCFCFG_FSTATn:RDY bit goes high. TCFCFG_FSTATn:RDY indicates that a Flash write or erase has completed and Flash memory is ready to accept a new command.
- When both the Hangup Interrupt flag (TCFCFG_FSTATn:HANGINT) and corresponding Interrupt Enable bit (i.e. TCFCFG_FICTRLn:HANGIE) are set. TCFCFG_FSTATn:HANGINT is set when the TCFCFG_FSTATn:HANG bit goes high. TCFCFG_FSTATn:HANG indicates that Flash memory has detected a Hangup-1 condition.
- When both the SEC interrupt flag (TCFCFG_FSECIR:SECINT) and corresponding Interrupt Enable bit (i.e. TCFCFG_FSECIR:SECIE) are set. TCFCFG_FSECIR:SECINT is set when the ECC checker logic has detected a single bit error in the Flash read data.

Bus error response

The Flash memory interface generates a bus error response under the following conditions:

- An ECC double or multi bit error condition
- Unprivileged write or erase access to Flash memory, and write access without the TCFCFG_FCFGR:WE bit being set to '1'
- 8- or 16-bit write access with the ECC enabled
- 8- or 16-bit write access during the second write sequence for a 32-bit write
- Unsupported access size (i.e. Flash read access greater than 64 bits, Flash write access greater than 32 bits and Flash configuration read/write access greater than 64 bits
- Unaligned write access to Flash memory
- TCM read access to the Flash macro when the auto algorithm is in progress
- Read/write/execute access to a protected sector
- Access to unused address space in Flash memory and the configuration registers
- Unprivileged write access to the configuration registers
- Writing to the TCFCFG_FECCCTRL register more than once
- Any deviation in the Flash configuration unlock sequence

9.4 Starting the Flash memory automatic algorithm

Writing to and erasing data in Flash memory is performed by launching the Flash memory's own automatic algorithms. The following commands are available: read/reset, write, macro erase and sector erase. The erase suspend and resume function is available during sector erase.

Command sequence table

Automatic algorithms for Flash memory write/erase are launched by writing one to six bytes, or half-words or words in succession (according to Table 9-14) to the Flash memory.

Command data must be written to the lower byte (with the exception of the write command program data). Data written to the upper bytes (in the case of half-word or word access) are ignored.

The data width used for the 4th bus write cycle of the write command (Program Address (PA) and Program Data (PD)) determines the write mode of the Flash (byte, half-word or word).

Table 9-14. Command sequence table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/reset	1	cmd_addr0	0xF0	-	-	-	-	-	-	-	-	-	-
Write/ program	4	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0xA0	PA	PD	-	-	-	-
Macro erase	6	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0x80	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0x10
Sector erase	6	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0x80	cmd_addr0	0xAA	cmd_addr1	0x55	SA	0x30
Sector erase suspend	1	SA	0xB0	-	-	-	-	-	-	-	-	-	-
Sector erase resume	1	SA	0x30	-	-	-	-	-	-	-	-	-	-

Table 9-15. Command address values for CPU and FPP modes

			Dual Flash devices		Single Flash devices
Operation mode	Command address	Small sector access	Big sector access (Flash-A)	Big sector access (Flash-B)	Big sector access (Flash-A)
CPU mode	cmd_addr0	mmmm_S550	mmmm_2AA0	mmmm_2AA8	mmmm_S550
	cmd_addr1	mmmm_SAA8	mmmm_1550	mmmm_1558	mmmm_SAA8
FPP mode	cmd_addr0	mmmm_mAA8	mmmm_mAA8	mmmm_mAA8	mmmm_mAA8
	cmd_addr1	mmmm_m554	mmmm_m554	mmmm_m554	mmmm_m554

Table 9-16. Write/Program sequence example to a program address in Sector SA2 (CPU mode)

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data
Write/ program	4	0x017F_5550	0xAA	0x017F_4AA8	0x55	0x017F_5550	0xA0	PA	PD

Notes:

- The addresses in Table 9-15 are the values in the CPU memory map. All addresses and data are represented using hexadecimal notation
- S [3:1] should point to the target sector in which the command is intended to operate
- S [0]: '1' for cmd_addr0 and '0' for cmd_addr1
- X represents an arbitrary value ('0'/ '1')
- PA stands for Program Address
- PD stands Program Data
- SA stands for Sector Address
- The addresses mmmm must point to the Flash memory module target address region (Flash A/ B) in which the command is intended to operate
- CPU/Flash data bits [31:8] are ignored during command writing (except for PD)
- In the 4th write cycle of the ‘write/program’ sequence, the PA should be the actual write location and PD is the program data*1 to be written
- The PA should be aligned*2 as a 4 byte grid (for a 32-bit write). If TCFCFG_FSTATn:WR32F = '0', the lower half-word (PD [15:0]) is written to Flash at address PA+0. If TCFCFG_FSTATn:WR32F = '1', the upper half-word and ECC (PD [31:16] & ECC [6:0]) are written to the incremented address PA + 2 If half-word and byte write are used, the ECC bits are not programmed. Therefore, it is recommended to use this only when the ECC is disabled to avoid errors at read back.
- Higher order bits of the PA are used to decode big or small sector accesses. In the case of big sector accesses, PA [3] is used to select the Flash macro while PA [16] is used to select the macro for small sector accesses. In cases where more than two macros exist, then PA[21] (row selection) and PA[3] (column selection) are used together to select the Flash macro for big sector accesses while small sectors accesses use PA [17:16])
- The SA in the table should point to the actual Sector Address in which the ‘sector erase’, ‘sector erase suspend’ or ‘sector erase resume’ command is intended to operate. Flash macro and sector selection is the same as that described for the PA in the previous bullet
- Writing an illegal address or data, or writing them in the incorrect order will reset the Flash memory to read mode
- It is possible to read data from the Flash between command write cycles. Command execution starts after the last write cycle
- Commands must not be written to sectors which are write protected. If a command/program sequence (i.e. cmd_addr0 or cmd_addr1) points to a write protected sector, the Flash memory will switch to read mode
- In cases where the program sequence is correct but the PA/SA points to a write protected sector, the Flash memory will enter a HANG state. Returning to the normal command state is achieved by implementing either a read/reset command, a software triggered Flash reset or a hard reset

*1: PD should be 32 bits wide if the ECC is enabled; otherwise it can be 8-, 16- or 32 bits wide

*2: For half-word accesses, the PA should be aligned as a 2-byte grid

Flash Parallel Programming

- In FPP, the Flash is addressed directly (i.e. no address translation is required) using the upper bits also used in Flash macro selection
- Therefore, 'mmmm' in Table 9-15 can point to any of the Flash address ranges shown in Figure 9-19

- In the 4th write cycle of the ‘write/program’ sequence, the PA is the write address. Even addresses can be specified in half-word mode only, while both even and odd addresses are permitted in byte mode. PD (i.e. write data) can be 8- or 16-bit wide
- In FPP, the ECC pins are also directly mapped to the external pins, and ECC calculation and checking are handled by the Flash programmer tool

9.5 Confirming the automatic algorithm execution state

The Flash memory performs the write/erase sequence via automatic algorithms. Flash memory hardware informs the outside world when internal operations have completed.

Hardware sequence flags

While the automatic algorithm is running, a read access to the Flash memory returns the status of the hardware sequence flags instead of Flash memory data. These 16-bit flags can be used to check the current status of the 'write/erase' operation.

The Flash interface internally replicates the hardware sequence flags four times in order to get 64- bit read data DQ [63:0]. As the flags are replicated, it doesn't actually matter where access is made in the Flash memory because the same information is read from the PA, PA + 2, PA + 4 or PA + 6. This would allow software to read back toggle bits from the same address wherever data was programmed to.

The hardware sequence flags consist of five bit pairs: DQ[7,15], DQ[6,14], DQ[5,13], DQ[3,11] and DQ[2,10], each of which has a defined function:

- DQ[7,15]: Data polling of the inverse of the value programmed (DATA[7,15])
- DQ[6,14]: Toggle bit 1 - inverts with each read when the automatic algorithm is in progress
- DQ[5,13]: Timing limit of the automatic algorithm has been exceeded
- DQ[3,11]: Command timeout for 'sector erase'
- DQ[2,10]: Toggle bit 2 - inverts with each read that accesses the erase suspended sector

Table 9-17. Bit assignment of hardware sequence flags

Read data bit no.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	DQ2	-	-
Read data bit no.	15	14	13	12	11	10	9	8
Hardware sequence flag	DQ15	DQ14	DQ13	-	DQ11	DQ10	-	-

Notes:

- The values of the bits DQ[4,12], DQ[1,9] and DQ [0,8] are not defined. Software must be written to tolerate any value assigned to these bits
- In the case of a 32-bit write, the hardware sequence flags will reflect the status of the lower half- word for first write sequence followed by the status of upper half-word for the 2nd write sequence
- Each byte has its own independent data polling flag to check if programming has completed (i.e. data changes from an inverted to a non-inverted value using DQ7/DQ15). The other hardware sequence flags reflect the Flash status and have the same information independent of the byte position from which they are read
- Reading the Flash hardware sequence flags can be done independently of the setting of the Flash write enable signal (TCFCFG_FCFGR:WE)
- As hardware sequence flags are not ECC protected, the ECC check is disabled when the automatic algorithm is in progress. Thus reading the algorithm status over TCM is not supported
- Even though 9.5 Confirming the automatic algorithm execution state and 9.6 Writing to and erasing Flash memory refer to the TCFLASH registers, they are also applicable to EEFLASH_IF, in which case the respective register bits should be considered
- To check whether automatic writing or erasing is being executed, either the hardware sequence flags or the TCFCFG_FSTATn:RDY bit can be read. More detailed information, however, is given by the hardware sequence flags
- The Flash memory will switch to the command state once the automatic algorithm has completed, which must be confirmed by either TCFCFG_FSTATn:RDY or the hardware sequence flags before performing the next operation. Additionally, the hardware sequence flags can be used to confirm whether the second or subsequent sector erase code write is valid. The function of all flags in each Flash state is shown in Table 9-18.

Table 9-18. Reading hardware sequence flags

State		DQ [7,15]	DQ [6,14]	DQ [5,13]	DQ [3,11]	DQ [2,10]	HANG[4]
Hard reset		DATA[1] [7,15]	DATA [6,14]	DATA [5,13]	DATA [3,11]	DATA [2,10]	0
Normal command state		DATA [7,15]	DATA [6,14]	DATA [5,13]	DATA [3,11]	DATA [2,10]	0
Program state		/DATA[2]	T[3]	0	0	0	0
Macro erase state		0	T	0	1	T	0
Command timeout state		0	T	0	0	T	0
Sector erase state		0	T	0	1	T	0
Erase suspend state[5]	Read on erase sector	0	0	0	1	T	0
	Read on non erase sector	DATA [7,15]	DATA [6,14]	DATA [5,13]	DATA [3,11]	DATA [2,10]	0
Hangup-1 state[6]	Program	/DATA	T	1	0	0	1
	Macro erase	0	T	1	1	T	1
	Sector erase	0	T	1	1	T	1

1. DATA in the table indicates the actual data read from the Flash memory
2. /DATA in the table indicates the inverse of bit [7] in each byte of the PD (i.e. /DATA is the inverse of PD [7,15])
3. T indicates that the flag is toggled on every read access
4. The TCFCFG_FSTATn:HANG status bit indicates whether or not the Flash is in the Hangup-1 state. (refer to 9.2.7 Flash Status Register (TCFCFG_FSTATn)) If a read is issued during the erase suspend state, the status of the hardware sequence flags will be output even if the sector erase operation was for an interleaved sector of the same Flash macro. In other words, trying to read SA23 during a sector erase operation of SA22 (in the erase suspend state) will output the status of hardware sequence flags and not the data stored in SA23
5. The value of the DQ flags depends on the cause of the Hangup-1 state i.e. program, macro erase or sector erase

Note: Hardware sequence flags toggle their state only in case of a read access to the corresponding Flash macro.

9.5.1 Data polling flags (DQ[7,15])

The data polling flags (DQ[7,15]) indicate if the automatic algorithm (write or erase) is being executed or has terminated.

Data polling flags (DQ[7,15])

The function of the DQ[7,15] flags in each Flash state is described in Table 9-18.

Write

A read access during the execution of the automatic write algorithm causes the Flash memory to output the inverted values of bits [7] and [15] of the PD (DATA:7 and DATA:15) regardless of which address (of the respective Flash macro) is being read.

A read access after termination of the automatic write algorithm is handled as a regular Flash read access and returns bits [7] and bit [15] (DATA:7 and DATA:15) of the Flash cell currently addressed. Hence to correctly identify the termination of a write command, polling should always be performed from the memory location that is being programmed.

Half-word and byte writing

A half-word write command writes data to the upper byte (DATA[15:8]) and lower byte (DATA[7:0]). DQ7 shows the inverted value of DATA[7] and DQ15 shows the inverted value of DATA[15].

A byte write command to an even address writes data to the lower byte (DATA[7:0]) only. DQ7 shows the inverted value of DATA[7]. A byte write command to an odd address writes data to the upper byte (DATA[15:8]) only. In this case DQ15 shows the inverted value of DATA[15].

Macro/sector erase

A read access during execution of the automatic erase algorithm causes the Flash memory to output a '0' regardless of which Flash address is being read. A read access after termination of the automatic erase algorithm is handled as a regular Flash read access and returns the data of the currently addressed Flash cell. Accessing an erased cell returns '1'.

Sector erase suspend

A read access during a ‘sector erase suspend’ state causes the Flash memory to output DQ15 = '0' and DQ7 = '0' if the address belongs to the sector being erased.

On the other hand, if the address does not belong to the sector being erased, the Flash memory outputs bits 7 and 15 (DATA: 7 and DATA:15) of the addressed memory cell.

These flags, together with the toggle bit flags DQ[6,14], enable a decision to be made as to whether the Flash memory is in the erase suspended state and which sector is being erased.

9.5.2 Toggle bit flags (DQ[6,14])

The toggle bit flags (DQ[6,14]), together with the data polling flags (DQ[7,15]), indicate if the automatic algorithm (write or erase) is being executed or has terminated.

Toggle bit flags (DQ[6,14])

The function of the DQ[6,14] flags in each Flash state is described in Table 9-18.

Write and macro/sector erase

Successive read accesses during execution of the automatic write or erase algorithm causes the Flash memory to toggle the DQ[6,14] flags for every read cycle, regardless of which address (of the respective Flash macro) is being read.

A read access after the automatic algorithm has terminated is handled as a regular Flash read access and returns bits 6 and 14 (DATA:6 and DATA:14) of the Flash cell currently addressed. Accessing an erased cell returns '1'.

Sector erase suspend

A read access during the ‘sector erase suspend’ state causes the Flash memory to output DQ6 = '0' and DQ14 = '0' if the address belongs to the sector being erased. On the other hand. if the address does not belong to the sector being erased, then bits 6 and 14 (DATA:6 and DATA:14) of the addressed memory cell are output.

Note: To detect the toggling state of the DQ [6,14] bits correctly, the same address (it need not be the write/erase address) should be read at least twice but preferably multiple times.

9.5.3 Timing-limit-exceeded flags (DQ[5,13])

The timing-limit-exceeded flags (DQ[5,13]) indicate that the execution of the automatic algorithm has exceeded the time (internal pulse count) prescribed in the Flash memory.

Timing limit exceeded flags (DQ[5,13])

The function of the DQ[5,13] flags in each Flash status is described in Table 9-18.

Write and macro/sector erase

A read access during execution of the automatic write or erase algorithm causes the Flash memory to output the status of the DQ[5,13] flags, regardless of which address (of the respective Flash macro) is being read.

A read access after the automatic write or erase algorithm has terminated is handled as a regular Flash read access and returns bits 5 and 13 (DATA:5 and DATA:13) of the Flash cell currently addressed.

The DQ[5,13] flags return '0' as long as the program or erase time remains within the prescribed time (maximum time required for write/erase). Otherwise the flags return '1' when read.

An unsuccessful write or erase operation can be determined if flags DQ[5,13] are '1' while flags DQ[6,14] and DQ[7,15] indicate that the automatic algorithm is still being executed.

For example, writing '1' to a Flash memory address where '0' has already been written causes a fail state. In response, the Flash memory locks and execution of the automatic algorithm will not terminate. As a result, valid data will not be output from the data polling flags DQ[7,15], the toggle bit flags DQ[6,14] will exceed the prescribed time limit and continue to toggle, which in turn will cause the timing-limit-exceeded flags DQ[5,13] to output '1'. This state is also indicated by the TCFCFG_FSTATn:HANG bit, which is set to '1'. When this state occurs, the ‘read/reset’ command must be executed.

Sector erase suspend

A read access in ‘sector erase suspend’ state causes the Flash memory to output DQ5 = '0' and DQ13 = '0' if the address belongs to the sector being erased.

If the address does not belong to the sector being erased, then bits 5 and 13 (DATA:5 and DATA:13) of the addressed memory cell are output.

9.5.4 Sector erase timer flags (DQ[3,11])

The sector erase timer flags (DQ[3,11]) indicate if the execution of the ‘sector erase’ command has been started or if the sector erase wait period is being applied, which then allows further ‘sector erase’ commands to be submitted.

Sector erase timer flags (DQ[3,11])

The function of the DQ[3,11] flags in each Flash status is described in Table 9-18.

After submitting a ‘sector erase’ command sequence, the sector erase wait period is applied. Within this period it is possible to submit further sector erase commands. The DQ[3,11] flags can be used to identify this wait period.

Sector erase

A read access during execution of the automatic erase algorithm causes the Flash memory to output the status of the DQ[3,11] flags, regardless of which Flash address is being read. A read access during the sector erase wait period returns '0'. After exceeding this wait period, the actual sector erase starts and a read access causes the DQ[3,11] flags to return '1'. Further sector erase commands will be ignored when DQ[3,11] are '1'.

A read access after the automatic erase algorithm has terminated is handled as a regular Flash read access and returns bits 3 and 11 (DATA:3 and DATA:11) of the Flash cell currently addressed. Accessing an erased cell returns '1'.

For more details about submitting multiple ‘sector erase’ commands refer to 9.6.4 Erasing optional data (sector erase).

The DQ[3,11] flags should be checked before and after submitting further ‘sector erase’ commands. If they return '1' after submitting the erase command, then this additional sector erase request may not be accepted.

Sector erase suspend

A read access in ‘sector erase suspend’ state causes the Flash memory to output DQ3 = '1' and DQ11 = '1' if the address belongs to the sector being erased.

If the address does not belong to the sector being erased, then bits 3 and 11 (DATA: 3 and DATA:11) of the addressed memory cell are output.

9.5.5 Toggle bit 2 flags (DQ[2,10])

The Toggle bit 2 flags (DQ[2,10]) indicate if a sector is in the ‘sector erase suspend’ state. It can also be used to check which sectors are to be erased.

Toggle bit-2 flags (DQ[2,10])

The function of the DQ[2,10] flags in each Flash status is described in Table 9-18.

The DQ[2,10] toggle bits can be used together with the DQ[6,14] toggle bits to determine whether the Flash is in the ‘sector erase suspend’ state or if the erase algorithm is currently being executed.

Sector erase

Successive read accesses during execution of the automatic sector erase algorithm causes the Flash memory to toggle the DQ[2,10] flags for every read cycle if the read address points to a sector which is to be erased. This can be used to determine if the sector currently being accessed belongs to sectors which are to be erased. For a ‘macro erase’ command, the DQ[2,10] flags toggle for all sectors. A read access to a sector which is not to be erased returns '1'.

A read access after the automatic algorithm has terminated is handled as a regular Flash read access and returns bits 2 and 10 (DATA:2 and DATA:10) of the Flash cell currently addressed. Accessing an erased cell returns '1'.

Sector erase suspend

Successive read accesses during a ‘sector erase suspend’ state causes the Flash memory to toggle the DQ[2,10] flags for every read cycle if the read address points to a sector which is to be erased. A read access to another sector returns bits 2 and 10 (DATA:2 and DATA:10) of the Flash cell currently addressed. Only ‘read’ and ‘sector erase resume’ operations are allowed during the ‘sector erase suspend’ state.

Both DQ[2,10] and DQ[6,14] flags are used for detecting an erase-suspended sector; the DQ[2,10] flags toggle, but DQ[6,14] do not.

Timing limit exceeded

When an unsuccessful ‘macro erase’ or ‘sector erase’ operation (Timing-limit-exceeded flag DQ[5,13] = '1') occurs, DQ[2,10] can be used to identify which sector caused the timeout state. In other words, DQ[2,10] will only toggle when accessing the sector which caused the timeout. For other sectors, '1' is returned.

9.6 Writing to and erasing Flash memory

This section describes the procedure for each Flash memory operation: read/reset; write; macro erase; sector erase; sector erase suspend; and sector erase resume.

Detailed explanation of Flash memory write/erase

By issuing a command sequence (refer to Table 9-14 in 9.4 Starting the Flash memory automatic algorithm) the Flash memory executes the automatic algorithm to perform ‘read/reset’, ‘write’ ‘macro erase’, ‘sector erase’, ‘sector erase suspend’ or ‘sector erase resume’ operations.

In between write accesses of the command sequence, other bus read/write cycles can be performed as long as no write access to the Flash memory, other than that defined by the command sequence, is performed. Even reading the Flash between write accesses is possible.

Termination of the automatic algorithm can be determined by polling either the hardware sequence flags, the RDY flag (TCFCFG_FSTATn:RDY) or the RDY interrupt flag (TCFCFG_FSTATn:RDYINT). A normal termination enables the Flash memory to return to the normal command state. An abnormal termination is determined by polling either the HANG flag (TCFCFG_FSTATn:HANG) or the HANG interrupt flag (TCFCFG_FSTATn_HANGINT).

In the following sections, each Flash memory operation is described in detail.

- 9.6.1 Setting read/reset state
- 9.6.2 Writing data by submitting the write command sequence
- 9.6.3 Erasing all data (macro erase)
- 9.6.4 Erasing optional data (sector erase)
- 9.6.5 Suspending sector erase
- 9.6.6 Sector erase resume

9.6.1 Setting read/reset state

This section describes the procedure for issuing the ‘read/reset’ command to set the Flash memory to the ‘read/reset’ state.

Read/reset command sequence

Table 9-19. Read/reset command sequence

Command sequence	Bus write access	1st bus write cycle	
		Address	Data
Read/reset	1	cmd_addr0	0xF0

Setting the Flash memory to the read/reset state

The ‘read/reset’ state is the initial state of the Flash memory. When the power is turned on and a command terminates normally, the Flash memory is set to the ‘read/reset’ state, allowing any command to be input.

In the timeout state (DQ [5] and DQ[13] = '1'), the Flash memory can be set to the ‘read/reset’ state by sending the read/reset command as given above (for details refer to Table 9-15 in 9.4 Starting the Flash memory automatic algorithm). Sector protection will most likely be implemented and to avoid any conflict (i.e. no write permission to another sector), choose cmd_addr0 in a way to point to the target sector of the intended write/ erase operation.

The ‘read/reset’ command is not required to read data using a regular read command. Instead it is mainly used to initialize the automatic algorithm when a command does not terminate normally.

The ‘read/reset’ command has no effect when the write or erase automatic algorithm is operating normally; it cannot be used to interrupt an ongoing ‘write’ or ‘erase’ command execution. Also resetting the Flash from the ‘sector erase suspend’ state is not possible.

9.6.2 Writing data by submitting the write command sequence

This section describes how to write data to the Flash memory using the ‘write/program’ command sequence.

Write command sequence

Table 9-20. Write command sequence

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Write/ program	4	cmd_add0	0xAA	cmd_add1	0x55	cmd_add0	0xA0	PA	PD	-	-	-	-

Starting the write automatic algorithm

The data write automatic algorithm of the Flash memory can be started by sending the ‘write/ program’ command sequence as given above (for details refer to Table 9-14 in 9.4 Starting the Flash memory automatic algorithm). To avoid conflict with probable sector protection (i.e. no write permission), write the sequence to the target sector in Flash memory. When the data write to the target address has completed in the fourth command cycle, the automatic algorithm for writing is started.

Specifying addresses

Any programming data is written to its target address, which is aligned to the respective target size. In the case of byte and half-word data writes, the direct address is given for the programmed Flash target location. In the case of 32-bit data writes, access must be repeated with the same aligned address. Internally, it is split into upper and lower half words.

Note:

- If ECC is enabled, only 32-bit write commands should be used. The Flash interface generates the ECC bits and correct addresses internally
- If ECC is disabled, 8-, 16- or 32-bit accesses can be used. However, for 32-bit data, repeated access is required
- Polling of the hardware sequence flag in the case of 32-bit data is done in two steps to take into account the upper and lower half words - the first polling is carried out from the original write address and polls the lower half word while the second has to be done from the write address + 2 to poll the upper half word

Notes on writing data

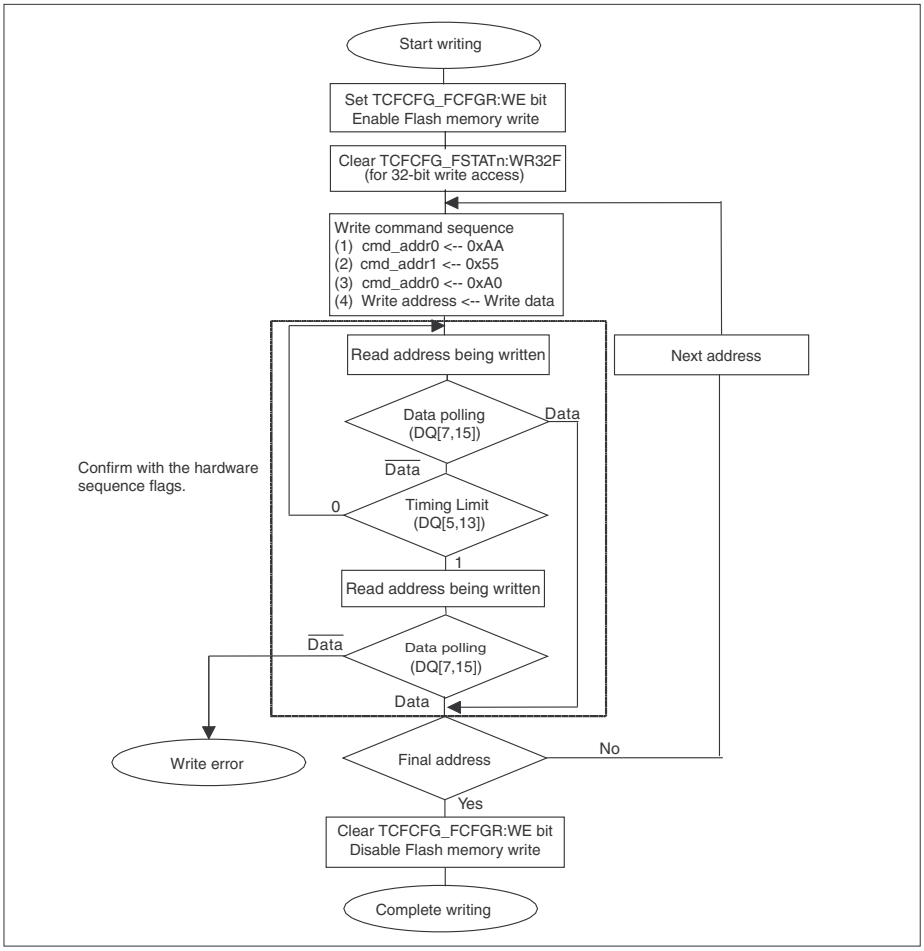
Writing cannot change an already set data bit in the Flash. In other words, trying to program a bit to ‘1’ which is already set to '0' causes the Flash memory to enter the 'Hangup-1' state. It is possible to identify whether or not the macro has entered the Hangup-1 state by looking at the TCFCFG_FSTATn:HANG status bit. A bit set to '0' can only be changed to '1' using the ‘erase’ operation.

All commands are ignored during execution of the automatic write algorithm. Asserting a CPU reset (power reset, external reset, software reset, watchdog reset or clock stop reset) or a software triggered Flash reset (i.e. TCFCFG_FCFGR:SWFRST) triggers a Flash hardware reset, which in turn cancels an ongoing write operation and switches the Flash memory to normal command state. In such cases the data of the written addresses is unpredictable.

Example for writing to the Flash memory

Figure 9-20 illustrates the procedure for writing to the Flash memory. The hardware sequence flags (9.5 Confirming the automatic algorithm execution state) can be used to determine the state of the automatic algorithm in the Flash memory. Here, the data polling flags (DQ[7,15]) are used to confirm that writing has terminated. Data polling must be performed on the memory location which is being programmed otherwise the transition of the DQ[7,15] flags at the termination of the write command cannot be detected.

Figure 9-20. Example of the Flash memory write procedure with data polling



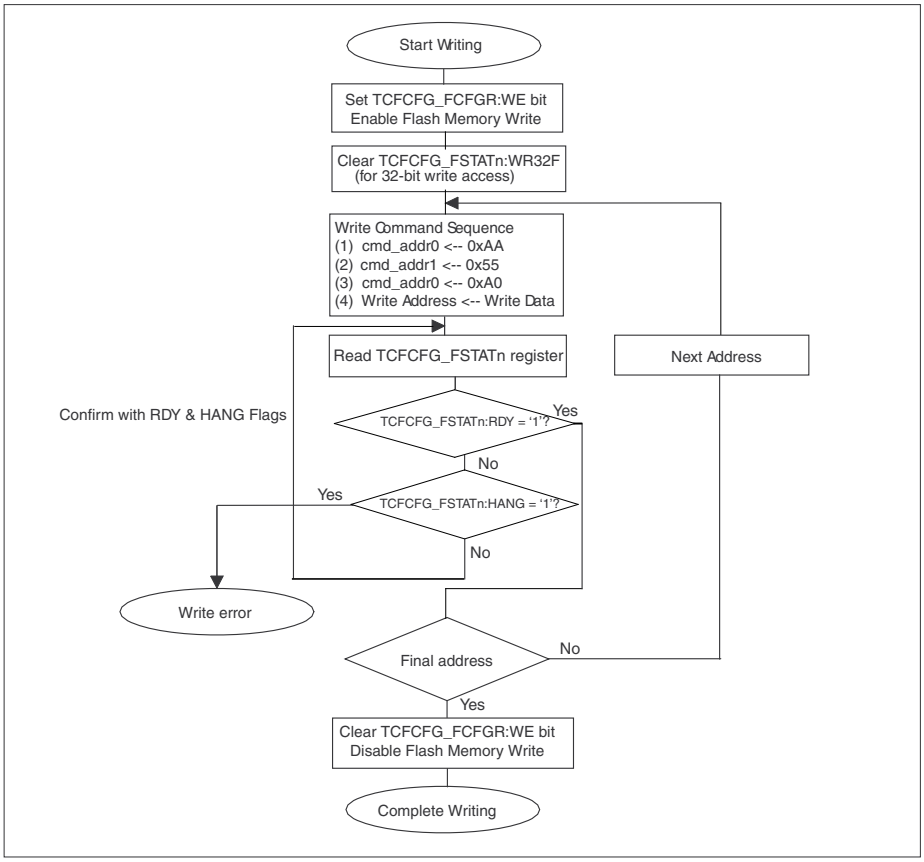
Note:

For more details on the values of cmd_addr0 and cmd_addr1, refer to Table 9-15. Each write sequence in the above flowchart should be repeated to complete the word write with ECC to Flash.

1. Write sequence for 32-bit data to PA + 0 (lower half-word data is written).
2. Data polling of 16-bit data from PA + 0 to check corresponding hardware sequence flags.
3. Write sequence for 32-bit data to PA + 0 (upper half-word data and ECC are written).
4. Data polling of 16-bit data from PA + 2 to check corresponding hardware sequence flags.

Alternatively, the TCFCFG_FSTATn:RDY and TCFCFG_FSTATn:HANG flags can be used to confirm Flash write completion, as illustrated in Figure 9-21.

Figure 9-21. Example of the Flash memory write procedure using the HANG and RDY flags



Note:

Refer to Table 9-15 for values of cmd_addr0 and cmd_addr1.

In the case of a 32-bit write with ECC, each write sequence in the above flowchart should be repeated as described in TCFLASH programming in CPU mode in 9.3 Operation of the TCFLASH

9.6.3 Erasing all data (macro erase)

This section describes the procedure for issuing the ‘macro erase’ command to erase all data in a Flash memory module.

Macro erase command sequence

Table 9-21. Macro erase command sequence

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Macro erase	6	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0x80	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0x10

Erasing all data in the Flash memory module (macro erase)

All data can be erased from a Flash memory module by continuously sending the ‘macro erase’ command sequence as given above (for more details refer to Table 9-15 in 9.4 Starting the Flash memory automatic algorithm) to any sector in the target Flash memory module.

The ‘macro erase’ command is executed in six bus operations. When the 6th cycle has completed, the ‘macro erase’ operation is started. For macro erase, the user does not need to write ‘0’ to the Flash memory before erasing. During execution of the automatic erase algorithm, the Flash memory automatically writes '0' before all the cells are erased. After the completion of a ‘macro erase’ all the memory cells store '1'.

Notes on macro erase

All commands are ignored during execution of the automatic erase algorithm. Asserting a Flash hardware reset by asserting either a CPU reset or a software triggered Flash reset (i.e. TCFCFG_FCFGR:SWFRST) triggers a Flash hardware reset, which in turn cancels an ongoing erase operation and switches the Flash memory to normal command state. In such cases the Flash macro data will be unpredictable. Erase access to a Flash macro with a write protected sector also asserts a Flash hardware reset and cancels the erase operation.

The Flash macro erase status can be found by either reading the hardware sequence flags DQ[7,15] and DQ[5,13] or the TCFCFG_FSTATn:RDY and TCFCFG_FSTATn:HANG flags as explained in previous sections.

9.6.4 Erasing optional data (sector erase)

This section describes the procedure for issuing the ‘sector erase’ command to erase optional data in the Flash memory. Individual sectors can be erased; multiple sectors can also be specified at the same time.

Sector erase command sequence

Table 9-22. Sector erase command sequence

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Sector erase	6	cmd_addr0	0xAA	cmd_addr1	0x55	cmd_addr0	0x80	cmd_addr0	0xAA	cmd_addr1	0x55	SA	0x30

Starting the sector erase automatic algorithm

Optional sectors in the Flash memory can be erased by continuously sending the ‘sector erase’ command sequence as given above (for details refer to Table 9-14 in 9.4 Starting the Flash memory automatic algorithm) to the target sector in the Flash memory.

Specifying sectors

A sector erase is initiated by submitting the sector erase command - which uses six bus write operations - to the target sector. After the last command (i.e 0x30) has been submitted, the sector erase wait time is applied for a minimum of 40 µs. To erase multiple sectors, write the erase code of the sixth cycle of the command sequence (i.e. 0x30) only to the next target sector within the wait time - the first five cycles do not have to be written.

During execution of the automatic erase algorithm, the Flash memory automatically writes '0' before all of the cells of target sector are erased. After the completion of the ‘sector erase’ command, all the memory cells of the target sector store '1'.

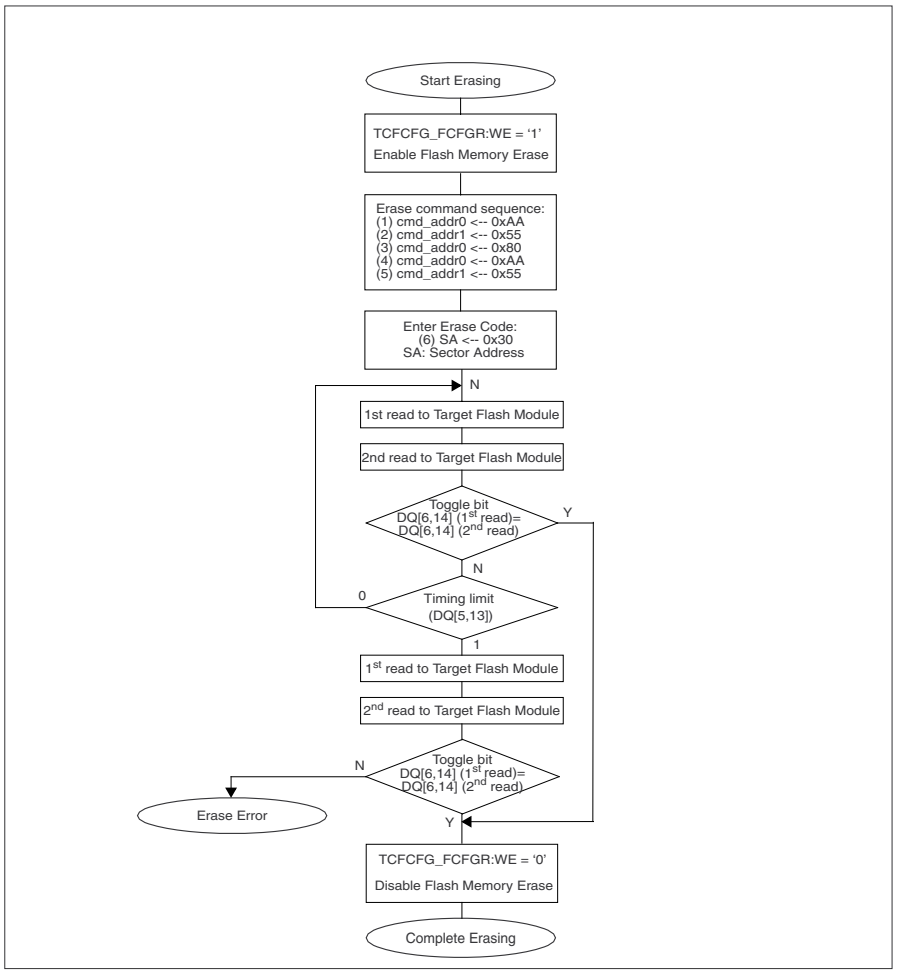
Notes on specifying multiple sectors

Each writing of the sector erase code restarts the sector erase wait period. The sector erase timer flags DQ[3,11] must be checked after submitting each erase code to ensure it has been accepted. An erase is started when the sector erase wait period of 40 µ**s** ends after the last sector erase code (0x30) has been written.

Example of erasing sectors in Flash memory

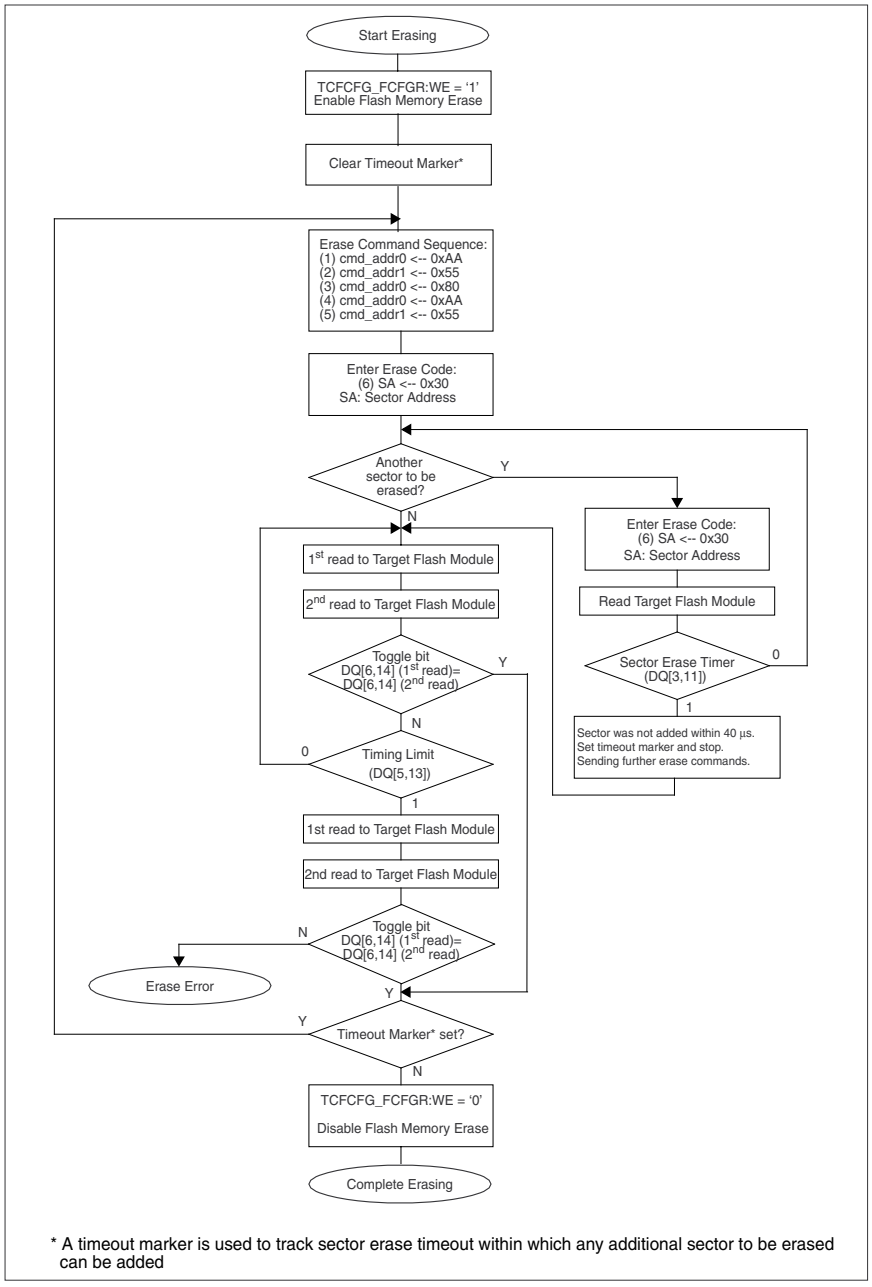
The hardware sequence flags (see 9.5 Confirming the automatic algorithm execution state) can be used to determine the state of the automatic algorithm in the Flash memory. Figure 9-22 shows the flow for a single sector erase in Flash while Figure 9-23 shows the flow for a multiple sector erase. In both examples the toggle bit flags DQ[6,14] are used to confirm that erasing has terminated. For the safe identification of the erase error state, the toggle bits DQ[6,14] must be checked again after reading DQ[5,13] = '1'. In the case of a multiple sector erase, the software should have a timeout marker to keep track of the sector erase wait time of 40 µs.

Figure 9-22. Flash memory single sector erase procedure



Note: For values of cmd_addr0 and cmd_addr1 refer to Table 9-15. Address 'SA' must point to the target sector for the sector erase.

Figure 9-23. Flash memory multiple sector erase procedure



9.6.5 Suspending sector erase

This section describes the procedure for issuing the ‘sector erase suspend’ command, which suspends the erasing of Flash memory sectors. In this state, data can be read from or written to sectors that are not to be erased.

Sector erase suspend command sequence

Table 9-23. Sector erase suspend command sequence

Command sequence	Bus write access	1st bus write cycle	
		Address	Data
Sector erase suspend	1	Sector Address (SA)	0xB0

Suspending erasing of Flash memory sectors

The erasing of Flash memory sectors can be suspended by sending the ‘sector erase suspend’ command (0xB0) to the target sector of the Flash memory module. Submitting this command suspends the sector erase operation being executed. Only ‘read’ and ‘erase resume’ operations are supported during the ‘sector erase suspend’ state. Further erase and program operations are forbidden.

The ‘sector erase suspend’ command is valid only while the sector erase operation is in progress or while the sector erase wait time is being applied. The command will be ignored during ‘macro erase’ or ‘write’ operations. If another ‘sector erase suspend’ command is issued during a ‘sector erase suspend’, the command will be ignored.

Entering the ‘sector erase suspend’ command during the sector erase wait period immediately terminates the wait period and cancels the requested erase operation. It takes a maximum of 20 μs until the Flash changes to the suspended state after the command has been submitted.

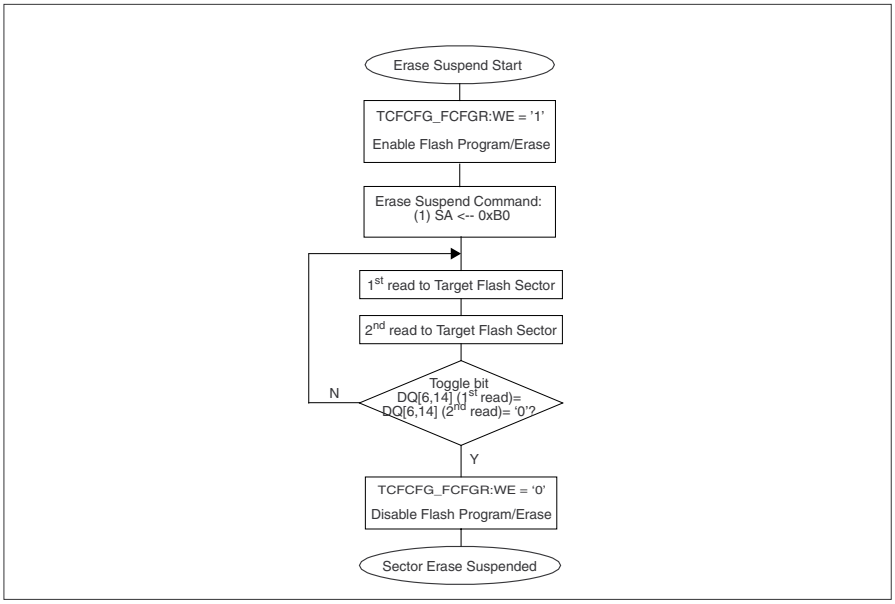
Notes on using the sector erase suspend function of the Flash memory

- Frequent ‘sector erase’ and ‘erase suspend’ commands may slow down the erase process or cause it to stick, which could then lead to the erase timeout limit (refer to device-specific datasheet for this timeout value), resulting in a Hangup-1 state.
- After restarting a sector erase, a minimum wait time of 2 ms must be applied before submitting a ‘sector erase suspend’ command.

Example of erasing sectors in the Flash memory

The hardware sequence flags (see 9.5 Confirming the automatic algorithm execution state) can be used to determine the state of the automatic algorithm in the Flash memory. It is possible to check whether or not the ‘sector erase suspend’ state is reached using the DQ[6,14] flags as shown in Figure 9-24. A read access in ‘sector erase suspend’ state causes the Flash memory to output both DQ6 and DQ14 = '0' if the address belongs to the sector being erased; normally these flags toggle in the ‘sector erase’ state.

Figure 9-24. Flash sector erase suspend procedure



Note: Address 'mmmm' should point to the target address region of the Flash memory module (either Flash-A or Flash-B) in which the command is meant to operate.

9.6.6 Sector erase resume

This section describes the procedure for issuing the ‘sector erase resume’ command to restart the erasing of Flash memory sectors.

Sector erase resume command sequence

Table 9-24. Sector erase resume command sequence

Command sequence	Bus write access	1st bus write cycle	
		Address	Data
Sector erase resume	1	SA	0x30

Resuming erasing of Flash memory sectors

The erasing of Flash memory sectors can be resumed after being suspended by sending the ‘sector erase resume’ command (0x30) to the target sector of Flash memory module. If a ‘sector erase resume’ command is issued during a sector erase, the command will be ignored.

The ‘sector erase resume’ command resumes the erasing of sectors from the sector erase suspend state.

9.7 Flash security

The scope of Flash security is to prevent the unauthorized read-out of Flash macros and access protection during application run time. Flash security is distributed between the TCFLASH_IF and Security Checker modules.

The Security Checker module provides sectorwise read, write and execute access permissions to the TCFLASH_IF based on the permission sets (primary and secondary) configured in the SDR[1] of the BootROM and based on the current active permission set[2] selected by the user. TCFLASH_IF checks all the read, write or execute accesses to the Flash memory against these settings and takes the necessary action described below.

Unauthorized access

- A read/execute access to a protected sector
  - The TCFLASH_IF prevents any access to the Flash and default data is output onto the bus
  - An external TCM error is signaled, which leads either to a data or prefetch abort in the case of CPU access via the TCM port, or a Slave Error Response (SLVERR) in the case of master access via the AXI slave interface
- A write/erase access to a protected sector
  - The Flash memory switches back to read mode if the program/command sequence points to a protected sector. If the program sequence is correct but the actual write address (i.e. PA/ SA) points to a protected sector, the Flash macro might enter a HANG state[3]
  - The Slave Error Response (SLVERR) is signaled in the case of master access via the AXI slave interface (this is not applicable for TCM as write access through TCM is not supported)

Sector and macro erase protection

- Sector erase is disabled for write protected sectors
- In CPU mode, Flash ‘macro erase’ is disabled if at least one Flash sector is write protected
- In FPP mode, Flash ‘macro erase’ is disabled if the chip erase disable marker (inside the SDR) is written with a specific value (all other values allow the ‘macro erase’ operation). If, however, chip erase is disabled, any write operation to the Flash will be discarded

Special permissions for BDR and SDR

Sectors storing BDR and SDR will have different and special access permissions, which are stored in different sectors.

Notes:

1. For configuration details concerning the Flash sectorwise access permissions, refer to the SDR layout described in 16. Boot ROM Software Interface
2. For active permission set selection details, refer to the TCFLASH Permission User Key Register (SCCFG_TCFPUSRKEY0~3) in 5. Security Checker.
3. When the Flash macro enters a HANG state, it can be returned to the normal command state by either a ‘read/reset’ command, a software triggered Flash reset (i.e. TCFCFG_FCFGR:SWFRST) or by a hard reset.

9.8 Notes on using Flash memory

This section contains notes on using Flash memory.

Input of a CPU reset

A CPU reset (power reset, external reset, software reset, watchdog reset or clock stop reset) asserts the Flash hardware reset. If this occurs during Flash writing, the data being written will be undefined.

Resetting the device once execution of a ‘sector erase’ command has begun corrupts the data in the sector. Restart the erase on this sector and allow it to complete.

Register configuration

Since register configuration and Flash memory access are carried out through different bus interfaces, it is recommended that the Flash Configuration Register (TCFCFG_FCFGR) settings are not changed while Flash memory access is in progress.

A Flash reset is asserted for a minimum of 500 ns. In the event of a software triggered Flash reset, the software should then wait until the Flash reset has completed before starting memory access. To know if the Flash is ready for the next command, the software can read the TCFCFG_FSTATn:RDY bit, which returns '1' after a period that does not exceed 80 μs.

Also the CAM output from the Flash macro is not valid during reset assertion and 70 ns after reset release. Thus it should be ensured that the TCFCFG_FCAMLRn and TCFCFG_FCAMHRn registers are not accessed during this period.

Program access to Flash memory

While data is being erased from or written to the Flash memory, reading is not possible. When it does happen that data or code needs to be read from the Flash memory while the memory is being erased or written to, the required data or code must first be copied to RAM before starting the erase/write operation. If more than one Flash module is available, then it is possible to perform erase/write commands on one Flash module while reading code and/or data from the other without having to copy data to RAM. Additionally, the ability to suspend a ‘sector erase’ operation means that data can be read from other sectors during the suspended state.

Ensure that the CPU is not executing code from a Flash memory that is being erased or written to. Instead, ensure that the code is executed from the other Flash memory (if available) or from the RAM. In devices with two Flash modules, big sectors of Flash-A and Flash-B are interleaved. If, for example, a write/erase is performed on the big sector region of Flash-A, then code execution should only be carried out from the small sector area of Flash-B and vice-versa.

Flash programming and hardware sequence flags

- Prefetch access to the interleaved address is disabled during Flash programming (i.e. TCFCFG_FCFGR:WE = '1'), to avoid that the toggle flags and prefetch disturb each other. Thus to retain the performance benefit of interleaved access, it is recommended that the software sets the TCFCFG_FCFGR:WE bit only before Flash programming and clears it after completion of the auto algorithm. The TCFCFG_FCFGR:WE bit must remain asserted during polling of the hardware sequence flags
- Since the hardware sequence flags are not ECC protected, the ECC check is disabled when the Flash auto algorithm is in progress. The ECC check is also disabled in the case of a 32-bit write until both write sequences (to cover the lower and upper 16 bits and the ECC) are completed because ECC data will not be consistent after the first write
- The TCFCFG_FSTATn:RDY flag or the TCFCFG_FSTATn:RDYINT interrupt should be used to know the auto algorithm status for a read protected sector
- Take care that concurrent accesses from different masters are avoided during Flash programming. Also it is recommended that the software ensures that all pending read accesses (i.e. already issued read accesses) are completed before starting a Flash write/erase operation

Flash performance

Due to design constraints, Flash-B small sector access through the AXI interface has an extra clock cycle latency at lower clock frequencies. Therefore, it is recommended not to put performance critical applications into the small sector region of Flash-B.

ECC error handling

- ECC checking is done on a 64-bit data read from the Flash (with two 32-bit wide ECC checks running in parallel) irrespective of access size
- In case of an ECC single bit error, the 64-bit aligned address is stored in TCFCFG_FECCEAR, making the position of the error uncertain within that 64-bit word

10. EEPROM Emulation Flash

This chapter explains the function and operation of the EEPROM Emulation Flash module and is applicable to devices that do not implement SHE support

10.1 Outline of the EEPROM Emulation Flash

This section describes the features and the block diagram of the EEPROM Emulation Flash module.

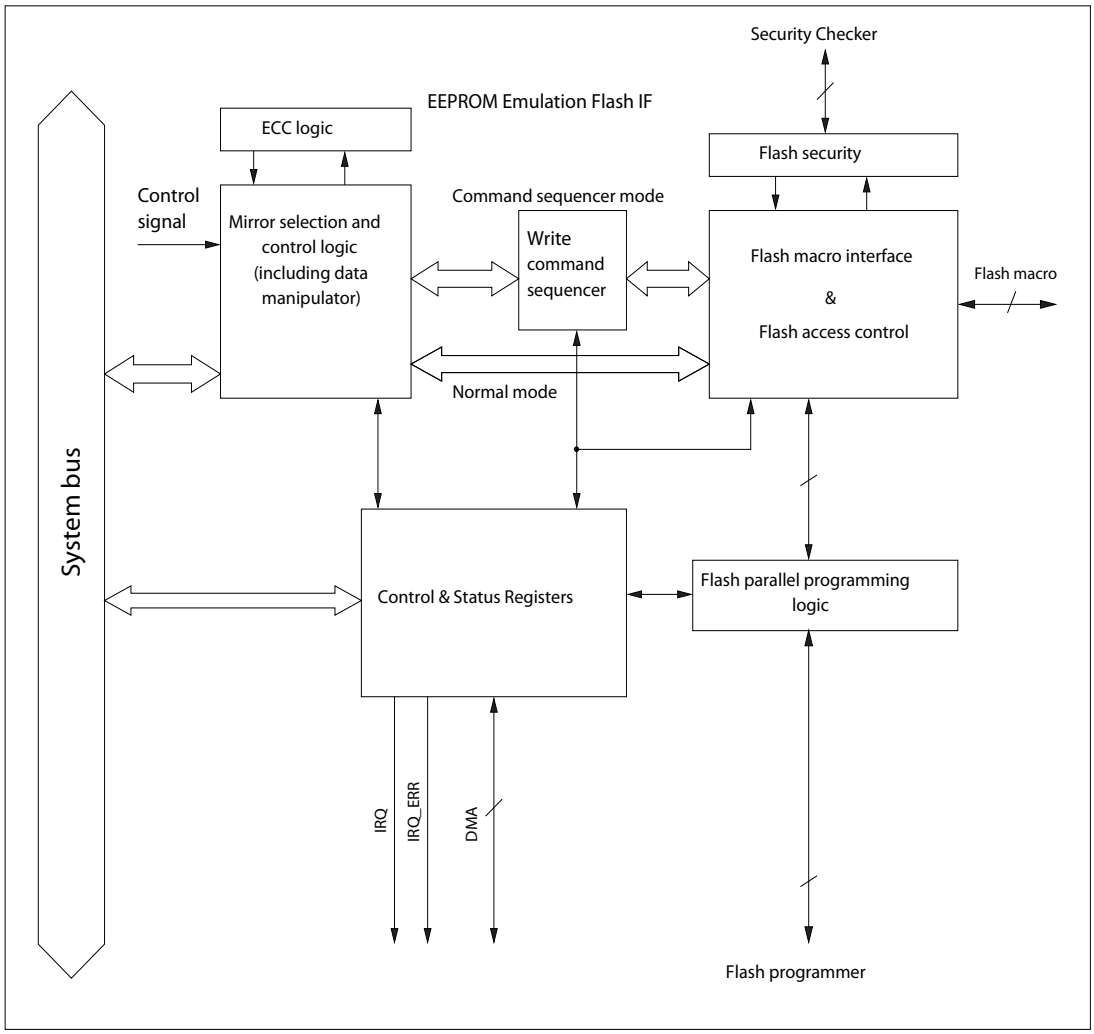
Features of EEPROM Emulation Flash

The EEPROM Emulation Flash memory is used to store data. EEPROM Emulation Flash write or erase can be done using instructions from the Central Processing Unit (CPU) or any other master via the EEPROM Emulation Flash Interface. The EEPROM Emulation Flash can also be programmed by the Flash Parallel Programmer. Other features of the EEPROM Emulation Flash are:

- It supports Flash read by the CPU in 8-,16-, 32- and 64-bit wide
- Flash write by the CPU is restricted to only 32-bit access if ECC is enabled. If ECC is disabled, it can be of 8-, 16- or 32-bit access
- Flash programming can be done either in direct mode (Flash Parallel Programming mode) or in CPU mode (normal mode or write command sequencer mode)
- It supports data programming by Direct Memory Access (DMA)
- It supports an ECC scheme for Single-Bit Error Correction and Double-Bit Error Detection (SEC-DED)
- ECC logic can be switched off using a register bit
- It supports ECC test by injecting errors
- Write access to the Flash Configuration Register settings are key protected
- It supports sector-wise read and write access permissions
- Flash write or erase is possible in MCU privileged mode and non-privileged mode
- Flash write or erase to a protected sector resets the Flash memory
- Read access to a protected sector outputs default data
- Access to an unused address range of Flash returns undefined data and generates a bus error response
- It detects the completion of write or erase commands using CPU interrupts

Block diagram of EEPROM Emulation Flash Interface

Figure 10-1. Block diagram of EEPROM Emulation Flash Interface



10.2 EEPROM Emulation Flash Memory Interface registers

This section describes the registers of the EEPROM Emulation Flash Memory Interface.

Registers of the EEPROM Emulation Flash Memory Interface

The following registers are available for the EEPROM Emulation Flash:

- Configuration Protection Key Register (EEFCFG_CPR)
- Configuration Register (EEFCFG_CR)
- ECC Control Register (EEFCFG_ECR)
- Write Command Sequencer Configuration Register (EEFCFG_WCR)
- Write Command Sequencer Status Register (EEFCFG_WSR)
- Data Bit Error Injection Register (EEFCFG_DBEIR)
- ECC Bit Error Injection Register (EEFCFG_EEIR)
- Write Mode Enable Register (EEFCFG_WMER)
- Interrupt Control Register (EEFCFG_ICR)
- Status Register (EEFCFG_SR)
- SEC Interrupt Register (EEFCFG_SECIR)
- ECC Error Address Register (EEFCFG_EEAR)
- Module Identification Register (EEFCFG_MIR)
- Extra Mode Enable Register (EEFCFG_EMENR)
- Flash CAM Output Lower Register (EEFCFG_FCAMLR)
- Flash CAM Output Higher Register (EEFCFG_FCAMHR)

Memory layout of EEPROM Emulation Flash Memory Interface registers

Table 10-1. Memory layout of EEPROM Emulation Flash Memory Interface registers

Offset	+3	+2	+1	+0
0x00000000	EEFCFG_CPR 00000000 00000000 00000000 00000000			
0x00000004	reserved 00000000 00000000 00000000 00000000			
0x00000008	EEFCFG_CR 00000000 00000000 00000000 00000011			
0x0000000C	EEFCFG_ECR 00000000 00000000 00000000 00000000			
0x00000010	EEFCFG_WCR 00000000 00000000 00000000 00000000			
0x00000014	EEFCFG_WSR 00000000 00000000 00000000 00000000			
0x00000018	EEFCFG_DBEIR 00000000 00000000 00000000 00000000			
0x0000001C	EEFCFG_EEIR 00000000 00000000 00000000 00000000			

Table 10-1. Memory layout of EEPROM Emulation Flash Memory Interface registers

Offset	+3	+2	+1	+0
0x00000020	EEFCFG_WMER 00000000 00000000 00000000 00000000			
0x00000024	EEFCFG_ICR 00000000 00000000 00000000 00000000			
0x00000028	EEFCFG_SR 00000000 00000000 00000000 00000000			
0x0000002C	EEFCFG_SECIR 00000000 00000000 00000000 00000000			
0x00000030	EEFCFG_EEAR 00000000 00000000 00000000 00000000			
0x00000034	EEFCFG_MIR 00000000 00000000 00000000 00000000			
0x00000038	EEFCFG_EMENR 00000000 00000000 00000001 00000000			
0x0000003C	reserved 00000000 00000000 00000000 00000000			
0x00000040	reserved 00000000 00000000 00000000 00000000			
0x00000044	reserved 00000000 00000000 00000000 00000000			
0x00000048	EEFCFG_FCAMLR XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000004C	EEFCFG_FCAMHR 00000000 00000000 000XXXXX XXXXXXXX			

10.2.1 Configuration Protection Key Register (EEFCFG_CPR)

The Configuration Protection Key Register (EEFCFG_CPR) should be written with the correct key value (0xCF6DF1A5) to unlock write access to the Interface Configuration Register (EEFCFG_CR) or the ECC Control Register (EEFCFG_ECR) and the ECC Bit Error Injection Registers (EEFCFG_DBEIR & EEFCFG_EEIR). Access is locked again after the next write on the Advanced High-Performance Configuration Bus (AHB). An incorrect key value or writing to the EEPROM Emulation Flash configuration registers without unlocking or any deviation in the above protection sequence causes an error response on the slave bus and hence a data abort. This register should be written using 32-bit writes only.

Configuration Protection Key Register (EEFCFG_CPR)

Figure 10-2. Configuration Protection Key Register (EEFCFG_CPR)

EEFCFG_CPR																															
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**
CPR[31]	CPR[30]	CPR[29]	CPR[28]	CPR[27]	CPR[26]	CPR[25]	CPR[24]	CPR[23]	CPR[22]	CPR[21]	CPR[20]	CPR[19]	CPR[18]	CPR[17]	CPR[16]	CPR[15]	CPR[14]	CPR[13]	CPR[12]	CPR[11]	CPR[10]	CPR[9]	CPR[8]	CPR[7]	CPR[6]	CPR[5]	CPR[4]	CPR[3]	CPR[2]	CPR[1]	CPR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-2. Configuration Protection Key Register (EEFCFG_CPR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CPR	Configuration Protection Register This register protects the EEPROM Emulation Flash Configuration Register (EEFCFG_CR), the EEPROM Emulation Flash ECC Control Register (EEFCFG_ECR) and the EEPROM Emulation Flash ECC Error Injection Registers (EEFCFG_DBEIR and EEFCFG_EEIR) from being accidentally modified by software. Writing the correct key value (i.e. 0xCF6DF1A5) to this register unlocks write access to the EEFCFG_CR, EEFCFG_ECR, EEFCFG_DBEIR and EEFCFG_EEIR registers. Writing any other value to this register causes a data abort. Reading this register always returns 0xFFFFFFFF when write access to EEFCFG_CR, EEFCFG_ECR, EEFCFG_DBEIR and EEFCFG_EEIR is unlocked. Reading this register always returns 0x00000000 when write access to EEFCFG_CR, EEFCFG_ECR, EEFCFG_DBEIR, and EEFCFG_EEIR is locked (default).

10.2.2 Configuration Register (EEFCFG_CR)

The Configuration Register (EEFCFG_CR) controls wait-cycle settings for EEPROM Emulation Flash access. This register also contains the EEPROM Emulation Flash write access enable and the EEPROM Emulation Flash reset control bit. This register is writable only if access is unlocked by writing the correct key value to the EEFCFG_CPR register.

Configuration Register (EEFCFG_CR)

Figure 10-3. Configuration Register (EEFCFG_CR)

EEFCFG_CR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SWFRST	read0	read0	read0	read0	read0	read0	read0	WE	read0	read0	read0	read0	read0	read0	FAWC[1]	FAWC[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WPS1	R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 10-3. Configuration Register (EEFCFG_CR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	SWFRST	Software Triggered Flash Reset This bit is used to reset the Flash macro by software. It also resets the write command sequencer if it is enabled. '0': No effect '1': Trigger Flash macro and write command sequencer reset The time to reset the Flash macro is 500 ns. This time is counted with respect to the maximum operable frequency. Therefore, for slow frequencies it will always be more than required. Reading this bit returns '0'.
[15:9]	read0	-
[8]	WE	Flash Write Enable This bit enables/disables writing to the Flash. '0': Flash write is disabled (default) '1': Flash write is enabled Write access to Flash with this bit set to '0' results in a bus error response.
[7:2]	read0	-

Table 10-3. Configuration Register (EEFCFG_CR) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	FAWC	Flash Access Wait Cycle Control These bits define the number of wait cycles required for Flash read and write access. These bits should be set based on the system clock frequency and Flash access time. Read/write wait cycles. '00': 0 wait cycles '01': 1 wait cycle '10': 2 wait cycles '11': 3 wait cycles (default) The relationship between the wait cycle setting and clock frequency is given as: FAWC = (system clock frequency/Flash operating frequency) - 1

10.2.3 ECC Control Register (EEFCFG_ECR)

The ECC Control Register (EEFCFG_ECR) controls the disabling of the ECC logic. This register is writable only if access is unlocked by writing the correct key value to the EEFCFG_CPR register. This register is writable to only once; writing more than once leads to a bus error response.

ECC Control Register (EEFCFG_ECR)

Figure 10-4. ECC Control Register (EEFCFG_ECR)

EEFCFG_ECR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ECCOFF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-4. ECC Control Register (EEFCFG_ECR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ECCOFF	ECC Disable/Switch-Off This bit disables ECC generation and checking. '0': ECC generation/checking is ON (default) '1': ECC generation/checking is OFF **Note:** ECC generation and checking are disabled irrespective of the used mirror scheme. Therefore when EEFCF-G_ECR:ECCOFF is disabled, ECC generation and checking are disabled when reading from any mirror.

10.2.4 Write Command Sequencer Configuration Register (EEFCFG_WCR)

The Write Command Sequencer Configuration Register (EEFCFG_WCR) is used to configure the write command sequencer and DMA.

Write Command Sequencer Configuration Register (EEFCFG_WCR)

Figure 10-5. Write Command Sequencer Configuration Register (EEFCFG_WCR)

EEFCFG_WCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CSEN	read0	read0	read0	read0	read0	read0	read0	DMAEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp	R0	R0	R0	R0	R0	R0	R0	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-5. Write Command Sequencer Configuration Register (EEFCFG_WCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	CSEN	Command Sequence Enable '0': Disables command sequencer '1': Enables command sequencer
[7:1]	read0	-
[0]	DMAEN	DMA Mode Enable '0': Disables DMA mode '1': Enables DMA mode The EEPROM Emulation Flash Interface generates a DMA request when EEFCFG_WCR:DMAEN and EEFCFG_WCR:CSEN are set and EEFCFG_WSR:ST is '00'.

10.2.5 Write Command Sequencer Status Register (EEFCFG_WSR)

The Write Command Sequencer Status Register (EEFCFG_WSR) returns the status of the write command sequencer.

Write Command Sequencer Status Register (EEFCFG_WSR)

Figure 10-6. Write Command Sequencer Status Register (EEFCFG_WSR)

EEFCFG_WSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ST[1]	ST[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-6. Write Command Sequencer Status Register (EEFCFG_WSR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1:0]	ST	Write Command Sequencer Status bits '00': The write command sequencer is either disabled (EEFCFG_WCR:CSEN = '0') or in idle state (EEFCFG_WCR:CSEN = '1'). In idle state, the sequencer is ready to receive a new write command '01': The write command sequencer submits the write command to the Flash module '10': The write command sequencer checks the progress of the write command by checking the Flash RDY status and the Flash hardware sequence flags '11': The write command sequencer was disabled (EEFCFG_WCR:CSEN = '0') during Flash writing (either in state '01' or '10') Do not access the Flash (read or write) as long as the command sequencer is not in idle state.

10.2.6 Data Bit Error Injection Register (EEFCFG_DBEIR)

Data Bit Error Injection Register (EEFCFG_DBEIR) is used to inject errors into the data part of the EEPROM Emulation Flash read data to test the ECC logic. It is writable only if access is unlocked by writing the correct key value to the EEFCFG_CPR register.

Data Bit Error Injection Register (EEFCFG_DBEIR)

Figure 10-7. Data Bit Error Injection Register (EEFCFG_DBEIR)

EEFCFG_DBEIR																															
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**
DBEIR[31]	DBEIR[30]	DBEIR[29]	DBEIR[28]	DBEIR[27]	DBEIR[26]	DBEIR[25]	DBEIR[24]	DBEIR[23]	DBEIR[22]	DBEIR[21]	DBEIR[20]	DBEIR[19]	DBEIR[18]	DBEIR[17]	DBEIR[16]	DBEIR[15]	DBEIR[14]	DBEIR[13]	DBEIR[12]	DBEIR[11]	DBEIR[10]	DBEIR[9]	DBEIR[8]	DBEIR[7]	DBEIR[6]	DBEIR[5]	DBEIR[4]	DBEIR[3]	DBEIR[2]	DBEIR[1]	DBEIR[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-7. Data Bit Error Injection Register (EEFCFG_DBEIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DBEIR	Data Bit Error Injection This bit is used to inject errors into data bits of Flash read data to test the ECC logic. Error injection is done by inverting respective data bits of the Flash read data before passing it to the ECC checking logic. '0': No effect on the corresponding bit of the Flash read data (default) '1': Flips the corresponding bit of the Flash read data

10.2.7 ECC Bit Error Injection Register (EEFCFG_EEIR)

The ECC Bit Error Injection Register (EEFCFG_EEIR) is used to inject errors into ECC bits of the EEPROM Emulation Flash read data to test the ECC logic. It is writable to only if access is unlocked by writing the correct key value to the EEFCFG_CPR register.

ECC Bit Error Injection Register (EEFCFG_EEIR)

Figure 10-8. ECC Bit Error Injection Register (EEFCFG_EEIR)

EEFCFG_EEIR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	EEIR[6]	EEIR[5]	EEIR[4]	EEIR[3]	EEIR[2]	EEIR[1]	EEIR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-8. ECC Bit Error Injection Register (EEFCFG_EEIR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6:0]	EEIR	ECC Bit Error Injection This bit is used to inject errors into the ECC bits of Flash read data to test the ECC logic. Error injection is done by inverting the respective ECC bits of Flash read data before passing it to the ECC checking logic. '0': No effect on the corresponding ECC bit of the Flash read data (default) '1': Flips the corresponding ECC bit of the Flash read data

10.2.8 Write Mode Enable Register (EEFCFG_WMER)

The Write Mode Enable Register (EEFCFG_WMER) is used to enable/disable ECC calculation during write access.

Write Mode Enable Register (EEFCFG_WMER)

Figure 10-9. Write Mode Enable Register (EEFCFG_WMER)

EEFCFG_WMER																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AME	read0	read0	read0	read0	read0	read0	read0	MME
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp	R0	R0	R0	R0	R0	R0	R0	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-9. Write Mode Enable Register (EEFCFG_WMER) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	AME	Automatic Mode Enable bit during Write 0: Automatic ECC is disabled during write (default) 1: Automatic ECC is enabled during write Reading this bit gives the automatic mode status Bit AME has higher priority than bit MME in the event both are set.
[7:1]	read0	-
[0]	MME	Manual Mode Enable bit during Write 0: Manual ECC is disabled during write (default) 1: Manual ECC is enabled during write Reading this bit gives the manual mode status

10.2.9 Interrupt Control Register (EEFCFG_ICR)

The Interrupt Control Register (EEFCFG_ICR) contains the interrupt enable and clear bits.

Interrupt Control Register (EEFCFG_ICR)

Figure 10-10. Interrupt Control Register (EEFCFG_ICR)

EEFCFG_ICR																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ERRIC	HANGIC	RDYIC	read0	read0	read0	read0	read0	ERRIE	HANGIE	RDYIE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0WP1	R0WP1	R0	R0	R0	R0	R0	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-10. Interrupt Control Register (EEFCFG_ICR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10]	ERRIC	Error Interrupt Clear '0': No effect '1': Clears EEFCFG_SR:ERRINT bit (Error Interrupt Status bit) Reading this bit returns '0'.
[9]	HANGIC	Hang Interrupt Clear '0': No effect '1': Clears EEFCFG_SR:HANGINT bit (Hang Interrupt Status bit) Reading this bit returns '0'.
[8]	RDYIC	Flash Ready Interrupt Clear '0': No effect '1': Clears EEFCFG_SR:RDYINT bit (Flash Ready Interrupt Status bit) Reading this bit returns '0'.
[7:3]	read0	-
[2]	ERRIE	Error Interrupt Enable '0': Disables error interrupt on error interrupt line (default) '1': Enables error interrupt on error interrupt line
[1]	HANGIE	Hang Interrupt Enable '0': Disables hang interrupt on error interrupt line (default) '1': Enables hang interrupt on error interrupt line

Table 10-10. Interrupt Control Register (EEFCFG_ICR) bits

Bit Position	Bit Field Name	Bit Description
[0]	RDYIE	Flash Ready Interrupt Enable '0': Disables Flash ready interrupt (default) '1': Enables Flash ready interrupt

10.2.10 Status Register (EEFCFG_SR)

The Status Register (EEFCFG_SR) holds the status of the EEFCFG_SR:RDY, EEFCFG_SR:HANG and EEFCFG_SR:WDCYC EEPROM Emulation Flash memory outputs and their respective interrupt status.

Status Register (EEFCFG_SR)

Figure 10-11. Status Register (EEFCFG_SR)

EEFCFG_SR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ERRINT	HANGINT	RDYINT	read0	read0	read0	read0	read0	WDCYC	HANG	RDY
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R0	R0	R0	R0	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-11. Status Register (EEFCFG_SR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:11]	read0	-
[10]	ERRINT	Error Interrupt '0': Error condition has not occurred (default) '1': Error condition has occurred The Error Interrupt bit will be set under the following conditions: ■ When the write command sequencer is disabled during an ongoing operation ■ Any write error during an ongoing transfer ■ This is a read only bit; writing to it has no effect.
[9]	HANGINT	Hang Interrupt This bit is set at the rising edge of the Flash HANG output. It is cleared by writing '1' to EEFCFG_ICR:HANGIC (Hang Interrupt Clear bit). '0': Hang condition has not occurred (default) '1': Hang condition has occurred This is a read-only bit; writing to this bit has no effect.

Table 10-11. Status Register (EEFCFG_SR) bits

Bit Position	Bit Field Name	Bit Description
[8]	RDYINT	Flash Ready Interrupt This bit is set on the rising edge of the Flash RDY output. It is cleared by writing '1' to EEFCFG_ICR:RDYIC (RDY Interrupt Clear bit). '0': The rising edge of the Flash RDY output is not detected (default) '1': The rising edge of the Flash RDY output is detected (i.e. Flash write or erase operation is completed) This is a read-only bit; writing to this bit has no effect.
[7:3]	read0	-
[2]	WDCYC	Program Address or Data taking in Cycle Flag Output from the Flash Macro This bit indicates the status of the Flash macro address data cycle. 0: Flash macro is not in address data cycle 1: Flash macro is in address data cycle This bit indicates in the current cycle that the Flash macro is accepting write data and write address (after accepting the correct command sequence in the previous cycles).
[1]	HANG	Hang Status This bit reflects the status of (sampled) Flash macro 'HANG' output and indicates whether or not the Flash macro is in a hang state. 0: Flash macro is operating normally (default) 1: Flash macro is in a hang state This bit is set to '1' under any of the following conditions: • Writing '1' to a memory cell containing '0' • This is a read-only bit; writing to it has no effect.
[0]	RDY	Flash Ready Status This bit reflects the status (sampled) of the Flash RDY output. In other words, it indicates whether or not the Flash macro is ready to accept a new command. '0': Flash programming is in progress; only a read/reset or suspend command is accepted '1': Flash is returned from auto-algorithm (program/erase) and is ready for a new command (default) This is a read-only bit; writing to it has no effect.

10.2.11 SEC Interrupt Register (EEFCFG_SECIR)

The SEC Interrupt Register (EEFCFG_SECIR) contains the ECC single-bit error interrupt status and corresponding interrupt enable and clear bits.

SEC Interrupt Register (EEFCFG_SECIR)

Figure 10-12. SEC Interrupt Register (EEFCFG_SECIR)

EEFCFG_SECIR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SECINT	read0	read0	read0	read0	read0	read0	read0	SECIC	read0	read0	read0	read0	read0	read0	read0	SECIE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-12. SEC Interrupt Register (EEFCFG_SECIR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	SECINT	ECC Single Error Correction Interrupt This bit is set when a single-bit error is detected/corrected during ECC checking. It is cleared by writing '1' to EEFCFG_SECIR:SECIC (ECC Single Error Interrupt Clear bit). '0': ECC single error has not occurred (default) '1': ECC single error has occurred This is a read-only bit, writing to this bit has no effect. Setting of EEFCFG_SECIR:SECINT has higher priority than clearing it.
[15:9]	read0	-
[8]	SECIC	ECC Single Error Correction Interrupt Clear '0': No effect '1': Clears EEFCFG_SECIR:SECINT (ECC Single Error Interrupt bit) Reading this bit returns '0'.
[7:1]	read0	-

Table 10-12. SEC Interrupt Register (EEFCFG_SECIR) bits

Bit Position	Bit Field Name	Bit Description
[0]	SECIE	ECC Single Error Correction Interrupt Enable '0': Disables ECC Single Error Correction interrupt (default) '1': Enables ECC Single Error Correction interrupt

10.2.12 ECC Error Address Register (EEFCFG_EEAR)

The ECC Error Address Register (EEFCFG_EEAR) stores the address of the EEPROM Emulation Flash memory location where the most recent ECC single bit error has occurred. Software can use this register to discover the error location.

ECC Error Address Register (EEFCFG_EEAR)

Figure 10-13. ECC Error Address Register (EEFCFG_EEAR)

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

Table 10-13. ECC Error Address Register (EEFCFG_EEAR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EEAR	ECC Error Address Register These read-only bits contain the Flash address of where the most recent ECC single bit error has occurred. This address may be helpful in marking the memory location as a weak cell. Writing to these bits has no effect.

10.2.13 Module Identification Register (EEFCFG_MIR)

The Module Identification Register (EEFCFG_MIR) contains the Module Identification Number, revision and patch details of the EEPROM Emulation Flash Interface module.

Module Identification Register (EEFCFG_MIR)

Figure 10-14. Module Identification Register (EEFCFG_MIR)

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

Table 10-14. Module Identification Register (EEFCFG_MIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module Identification Number This register identifies the particular version of the hardware used in the device. The information contained in the register helps in the development of software for the particular hardware version. **Note:** For more details on the module ID number, refer to the device-specific datasheet.

10.2.14 Extra Mode Enable Register (EEFCFG_EMENR)

The Extra Mode Enable Register (EEFCFG_EMENR) contains the extra mode control bit.

Extra Mode Enable Register (EEFCFG_EMENR)

Figure 10-15. Extra Mode Enable Register (EEFCFG_EMENR)

EEFCFG_EMENR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AEE	read0	read0	read0	read0	read0	read0	read0	EMEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp	R0	R0	R0	R0	R0	R0	R0	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 10-15. Extra Mode Enable Register (EEFCFG_EMENR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	AEE	Arbitration Error Enable The default value for this bit is '1' (enabled). This bit can be disabled during wait states in which all buses may stall. In the worst case, a watchdog reset may occur.
[7:1]	read0	-
[0]	EMEN	Extra Mode Enable The software should always write this bit to '0'. Writing '1' leads to unknown behavior.

10.2.15 Flash CAM Output Lower Register (EEFCFG_FCAMLR)

The Flash CAM output lower register (EEFCFG_FCAMLR) stores bits[31:0] of the Flash macro CAM output.

Flash Register (EEFCFG_FCAMLR)

Figure 10-16. Flash CAM output lower register (EEFCFG_FCAMLR)

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

Table 10-16. Flash CAM output lower register (EEFCFG_FCAMLR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CAML	Flash CAM Output Lower Word This is a read-only register. Reading this register returns the lower word of the Flash macro CAM output. The CAM output gives the unique ID number of the Flash macro.

10.2.16 Flash CAM Output Higher Register (EEFCFG_FCAMHR)

The Flash CAM output upper register (EEFCFG_FCAMHR) stores bits[44:32] of the Flash macro CAM.

Flash Register (EEFCFG_FCAMHR)

Figure 10-17. Flash CAM output upper register (EEFCFG_FCAMHR)

EEFCFG_FCAMHR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CAMH[12]	CAMH[11]	CAMH[10]	CAMH[9]	CAMH[8]	CAMH[7]	CAMH[6]	CAMH[5]	CAMH[4]	CAMH[3]	CAMH[2]	CAMH[1]	CAMH[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 10-17. Flash CAM output upper register (EEFCFG_FCAMHR)

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:13]	read0	-
[12:0]	CAMH	Flash CAM Output Upper Word This is a read-only register. Reading this register returns the upper word of the Flash macro CAM output. The CAM output gives the unique ID number of the Flash macro.

10.3 Operation of the EEPROM Emulation Flash Interface

This chapter describes the operation of EEPROM Emulation Flash Interface.

EEPROM Emulation Flash memory

The EEPROM Emulation Flash macro has up to eight 8KB sectors (refer to the device-specific datasheet for details).

The features supported by the EEPROM Emulation Flash macro include:

- An automatic program algorithm used for write or erase
- A sector erase function (for any combination of sectors)
- Erase pause/resume functions
- The indication of write and macro erase command sequence detection
- The detection of write or erase completion using data polling or by CPU interrupts
- The detection of a hang state. For more details, refer to 9. Tightly Coupled Flash'
- A fast write mode (also known as 'unlock bypass mode') by sending a two-cycle write command. Refer to 10.4 Starting the EEPROM Emulation Flash memory automatic algorithm for more details
- Flash read access time with a minimum of one bus clock cycle

EEPROM Emulation Flash memory operation modes

The Flash can be operated in two different operation modes, CPU and Flash Parallel Programming.

- CPU mode: This mode can be handled in two ways, normal mode and write command sequencer mode. In both modes, the EEPROM Emulation Flash can be accessed by the CPU or any other master in the system via the EEPROM Emulation Flash Interface. ECC calculation is handled inside the EEPROM Emulation Flash Interface
  - Normal mode : This is the default mode of EEPROM Emulation Flash operation in which writing/erasing of the EEPROM Emulation Flash is performed by a programming access sequence to trigger its auto-algorithm for write, erase etc. For the list of command sequences supported and related information, refer to 10.4 Starting the EEPROM Emulation Flash memory automatic algorithm
  - Write command sequencer mode : This is a special mode of EEPROM Emulation Flash operation in which a write to the EEPROM Emulation Flash is performed without a programming access sequence (auto-algorithm for write is handled by the EEPROM Emulation Flash Interface itself) whereas an erase is performed by a programming access sequence to trigger its auto-algorithm
- Flash Parallel Programming mode: This mode is entered by setting mode pins MD [5:0] to 6'bxxx111 in board test mode. In this mode the device behaves as a standalone Flash. Flash interface pins are directly mapped to external pins so that the Flash can be accessed directly via the external MCU pins, and write or erase is performed using a Flash memory programmer. In this mode the ECC bits are treated equally to the data bits in Flash. Calculation of ECC is handled by the Flash programmer tool, thus no ECC calculation is done by the Flash interface

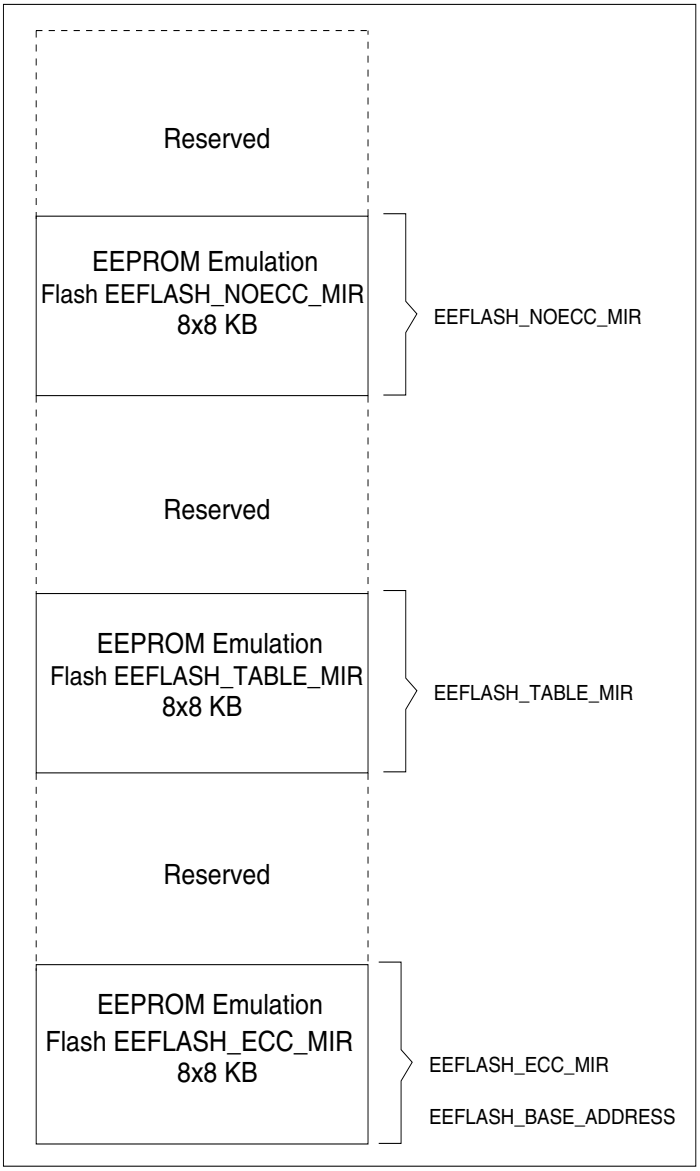
EEPROM Emulation Flash memory address/sector mapping

Address mapping of Flash sectors depends on the mode of operation.

- In CPU mode, Flash memory is reachable via three addresses (refer to Address mirroring and access control for more details). Address mapping is shown in Figure 10-18 and Figure 10-19
- Address mapping of Flash memory is such that all (8 KB) sectors are kept at a fixed position
- In Flash Parallel Programming mode[1], no address mapping is carried out. All address pins are directly connected to the Flash. Thus Flash addressing is just the physical Flash addressing as shown in Figure 10-20

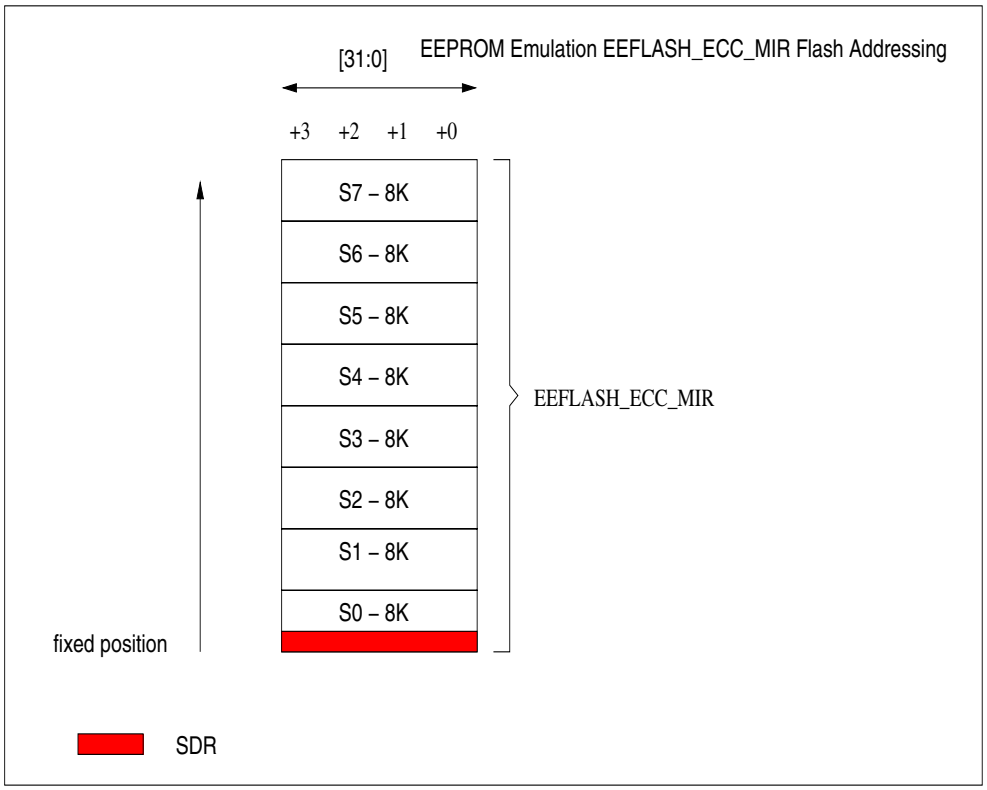
Note 1: To reduce the pin count requirement, Flash Parallel Programming mode supports only 32-bit reads.

Figure 10-18. EEPROM Emulation Flash address mapping



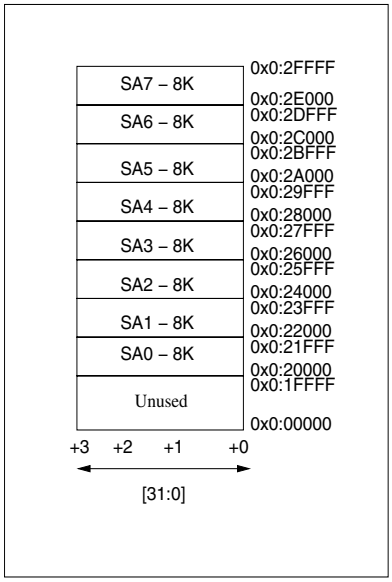
Note: EEFLASH_BASE_ADDRESS, EEFLASH_ECC_MIR,EEFLASH_TABLE_MIR and EEFLASH_NOECC_MIR are generally placed. Refer to the datasheet of a particular device to see the actual address.

Figure 10-19. Flash sector/address mapping - CPU mode



Note: Security Description Records (SDR) are located at the lower addresses of S0.

Figure 10-20. Flash sector/address mapping - Parallel Programming mode



EEPROM Emulation Flash programming in CPU mode

Flash write or erase by the CPU in normal mode should follow the command sequence table given in 10.4 Starting the EEPROM Emulation Flash memory automatic algorithm. Concerning Flash write or erase in write command sequencer mode, the Flash command sequence for the program is handled by the hardware itself.

Regarding ECC calculation and writing to Flash, Flash write in CPU mode (normal and write command sequencer mode) is restricted to 32-bit mode[2] only. ECC calculation enabling is handled differently for normal mode and write command sequencer mode.

Note 2:

If ECC is enabled (EEFCFG_ECR:ECCOFF = 0) and access is through EEFLASH_ECC_MIR, Flash write in CPU mode is restricted to 32-bit mode to allow ECC calculation. However, if ECC is disabled (EEFCFG_ECR:ECCOFF = 1), Flash write can be 8-, 16- or 32-bit wide irrespective of the Mirror used to access Flash memory. There is no true 32-bit write support by the Flash memory, but the interface accepts this data type through the above defined ECC calculation enabling. When the EEFCFG_ECR:ECCOFF bit is set to '0', then complete ECC is switched OFF. During EEFCFG_ECR:ECCOFF = '0', both the manual and automatic calculation of ECC enabling in write mode is also switched off irrespective of the concern bit settings.

Enabling ECC calculation in normal mode

In normal mode ECC calculation is manually or automatically enabled, depending on the value of the EEFCFG_WMER:MME (Manual ECC Mode Enable) and EEFCFG_WMER:AME (Automatic ECC Mode Enable) bits[3]. For more details, refer to the flowchart shown in Figure 10-22.

- Manual ECC enable
  - Send the Flash write command sequence with the Program Address (PA) and 32-bit data to be written (32 bits data must always be transferred to calculate the ECC from the complete data). At the end of the program command sequence (i.e. when EEFCFG_SR:WDCYC = 1), set EEFCFG_WMER:MME = 0. The lower 16-bit data is written to the Program Address and the upper 16 bits are ignored
  - Send the Flash write command sequence with the PA and 32-bit data to be written (32-bit data must always be transferred). At the end of the program command sequence (i.e. when EEFCFG_SR:WDCYC = 1), set EEFCFG_WMER:MME = 1. The upper 16-bit data plus ECC is written to the incremental address (PA + 2) and the lower 16 bits are ignored.
- Automatic ECC enable:
  - Set EEFCFG_WMER:AME = 1
  - Send the Flash write command sequence (as given in Table 10-18) with the Program Address and 32-bit data to be written (32-bit data must always be transferred in order to calculate the ECC). Repeat the process again after checking the EEFCFG_SR:RDY bit
  - At the end of each program command sequence, ECC enable is toggled automatically

During the first 32-bit write access, the ECC enable bit is '0' and the lower 16-bit data is written. During the second 32-bit write access, the ECC enable bit is '1' and the upper 16-bit data is written with ECC.

Note 3: For more details, refer to 10.2 EEPROM Emulation Flash Memory Interface registers.

Enabling ECC calculation in write command sequencer mode

- Send the Program Address and 32-bit data to be written (always 32-bit data must be transferred to calculate the ECC from the complete data)
- Two 16-bit writes to the Flash are handled by the command sequencer. The lower 16 bits are written without ECC and the upper 16 bits are written with ECC. ECC calculation is handled automatically by the interface, irrespective of the manual and automatic enable bits.

Address mirroring and access control

The EEPROM Emulation Flash is mapped to three different mirroring locations in the memory map[4]. Accessing EEPROM Emulation Flash through this mirroring area is as follows:

- EEFLASH_ECC_MIR: Depending on the EEFCFG_ECR:ECCOFF bit, read and write access by the CPU in this mirror region always allows ECC.
- EEFLASH_NOECC_MIR: This mirror allows read and write accesses without ECC, independent of the EEFCFG_ECR:ECCOFF and ECC calculation enable bits.

Note 4: Address mirroring is applicable for both the CPU and Write command sequencer mode.

Write command sequencer operation

The EEPROM Emulation Flash can be written to by two methods. Either in normal mode by manually submitting the write sequence as described in the 10.2.4 Write Command Sequencer Configuration Register (EEFCFG_WCR) (EEFCFG_WCR:CSEN = ’0’) or by using the write command sequencer (EEFCFG_WCR:CSEN = ’1’). After activating the write command sequencer, each CPU write access to the EEPROM Emulation Flash is handled as a write request to the Flash memory. The Flash interface prefixes each write data with the write command sequence. It is not possible to submit any other command sequence to the Flash as long as EEFCFG_WCR:CSEN is set to ‘1’.

Activating and deactivating the write command sequencer

After a reset, the write command sequencer is always disabled. It can be activated by setting EEFCFG_WCR:CSEN to ‘1’. This should be done only when the EEPROM Emulation Flash is in the read/reset or sector erase suspend state and no command sequence has been written to the Flash (not even the first write cycle of a new command sequence). Disabling the write command sequencer should only be done in an idle state (EEFCFG_WSR:ST[1:0] = ’00"). Disabling the write command sequencer in anything other than an idle state leads to an error response. Flash memory (macro) moves to an undefined state. Software reset (EEFCFG_CR:SWFRST = ‘1’) must be issued to reset the Flash memory.

Specifying addresses

Writing to the EEPROM Emulation Flash in write command sequencer mode is possible in 32-bit write accesses only due to the ECC calculation. With 32-bit writes, a 4 byte aligned address must be specified. The EEPROM Emulation Flash Interface calculates ECC for the 32-bit data and programs the 32-bit data to the Flash. Automatic command sequence and ECC enabling is handled by the interface itself.

Writing and reading the EEPROM Emulation Flash with activated write command sequencer

Writing and reading the EEPROM Emulation Flash is possible only when the write command sequencer is in an idle state (EEFCFG_WSR:ST[1:0] = "00"). Then 8-, 16-, 32- or 64-bit reads are possible. When the write command sequencer is not in an idle state, no Flash read access is executed and an error response is given on the bus.

DMA access using write command sequencer

The write command sequencer can only handle one write command at a time. A second write request is ignored and an error response given when a write operation is ongoing. The progress of a write operation can be checked by reading the write command sequencer state. Memory blocks can be transferred by using the DMA function. DMA block and burst mode are supported.

Note:

It is recommended to use block transfer mode with a block size of 32 bits in the event of a write. A DMA request is generated when: the EEFCFG_WCR:DMAEN bit is set to ’1’; the write command sequencer is enabled (EEFCFG_WCR:CSEN = ’1’); and the write command sequencer is in an idle state (EEFCFG_WSR:ST[1:0] = ’00’). Any write error during an ongoing transfer stops the DMA and issues an interrupt[*1].

1. DMA access can only be done by enabling the write command sequencer. Enabling DMA in other modes leads to an error response.

Flash access wait cycles

Based on the system clock frequency and Flash access time (refer to the datasheet for the access time value), the number of wait cycles required for Flash access can be configured by writing to the EEFCFG_CR:FAWC[1:0] bits. By default, these bits are configured for three (3) wait cycles. The same wait cycle settings are applicable for both read and write access to the Flash.

The read data from the Flash is valid only after the configured number of wait cycles. Once the wait states are configured, they continue to apply until they are changed by writing to the Configuration registers. For details on wait cycle settings, refer to Table 10-3.

ECC logic

- The ECC module uses Arm Cortex-R4's 32-bit ECC scheme for ECC encoding during Flash write and ECC syndrome decoding during Flash read
- During a write operation, the ECC module computes the ECC on 32-bit write data. The ECC can be calculated based on manual or automatic enabling. For more details, refer to 10.6 Notes on enabling ECC.
- During a read operation, the ECC syndrome is computed for the Flash read data syndrome. It is thus interpreted for no error, single error, double and uncorrectable errors. ECC for erase flash gives no error
- The ECC module also has an option to inject errors into both the data and the ECC bits read from the Flash to test ECC functionality
- There is an option to disable/switch-off ECC by setting a configuration register bit (i.e. EEFCFCG_ECR:ECCOFF)

Interrupts

The Flash memory interface generates interrupts under the following conditions:

- The Ready Interrupt Flag (i.e. EEFCFG_SR:RDYINT) is set when the Flash memory RDY output goes high (indicating that a Flash write or erase is completed and Flash memory is ready to accept a new command). The interrupt (IRQ) is generated when the EEFCFG_SR:RDYINT interrupt flag and the corresponding Interrupt Enable bit (i.e. EEFCFG_ICR:RDYIE) are set
- The Hang Interrupt Flag (i.e. EEFCFG_SR:HANGINT) is set when the Flash memory HANG output goes high (indicating that the Flash memory has detected a hang condition). The interrupt (IRQ_ERR) is generated when the EEFCFG_SR:HANGINT interrupt flag and the corresponding Interrupt Enable bit (EEFCFG_ICR:HANGIE) are set
- The SEC Interrupt Flag (i.e. EEFCFG_SECIR:SECINT) is set when the ECC checker logic has detected and corrected a single-bit error in the Flash read data. The interrupt (IRQ) is generated if EEFCFG_SECIR:SECINT interrupt flag and the corresponding Interrupt Enable bit (EEFCFG_SECIR:SECIE) are set
- The Error Interrupt Flag (i.e EEFCFG_SR:ERRINT) is set when any write error has been detected. The interrupt (IRQ_ERR) is generated if EEFCFG_SR:ERRINT and the corresponding Interrupt Enable bit EEFCFG_ICR:ERRIE are set

Bus error response

The EEPROM Emulation Flash memory interface generates a bus error response under the following conditions:

- An ECC double or multi-bit error condition
- A write access to Flash memory without the EEFCFG_CR:WE bit set to '1'
- 8- or 16-bit write access with ECC enabled
- Unaligned write access to Flash memory
- Read/write access to a protected sector
- Access to unused Flash memory and Configuration register address space
- Unprivileged write access to the configuration registers except EEFCFG_WCR and EEFCFG_EMENR
- Writing to the EEFCFG_ECR register more than once
- Any deviation in the Flash configuration unlock sequence
- EEFCFG_WCR:DMAEN is enabled when EEFCFG_WCR:CSEN is disabled
- In write command sequencer mode when the current access is ongoing and the next one is ignored

Flash security

To prevent unauthorized reading of the Flash macros and access protection during application, run time Flash security is introduced. Flash security is distributed between the Flash Interface and Security Checker modules.

The Flash Interface receives EEPROM Emulation Flash sector-wise read and write access permissions from the Security Checker module and performs the following tasks:

- Access is allowed to the addressed sector if permission is granted
- If read/execute access is denied to the addressed sector
  - The Flash Interface prevents access to the Flash and default data is output onto the bus
  - A bus error response is generated to inform the master
- If write access is denied to the addressed sector
  - The Flash memory falls back to read mode if the program/command sequence points to a protected sector. If, however, the program sequence is correct but the actual write address (i.e. PA/SA) points to a protected sector, the Flash macro might enter a HANG state
  - A bus error response is generated to inform the master
- Flash access is completely blocked during a scan test to avoid scanning out the Flash contents. This is done by forcing the Flash internal supply reset to '0' during a scan test mode, causing the Flash macro to enter sleep state

Note: All read, write or execute accesses to Flash memory are checked against the received permission settings.

Sector and macro erase protection

- Sector erase is disabled for a write protected sector
- In CPU mode, the Flash macro erase is disabled if at least one Flash sector is write protected.
- In FPP mode, the Flash macro erase is disabled if the chip erase disable marker (inside the SDR) is written with a specific value (all other values will allow the macro erase operation). Also if chip erase is disabled, any write operation to the Flash will also be discarded

Special Permissions for BDR & SDR

- Sectors storing a Boot Description Record (BDR) and SDR will have special and different access permissions and hence they are stored in different sectors

Notes:

- In command sequencer mode, a read permission to the respective sector should be enabled, otherwise an error interrupt will be generated
- Configuration details of Flash sector-wise access permissions can be found in the SDR layout of BOOTROM Software Interface (16. Boot ROM Software Interface).
- For active permission set selection details refer to the Security Checker module (5. Security Checker).
- When the Flash macro enters a HANG state, it can be brought back to normal command state by: a read/reset command; a software triggered Flash reset (i.e. TCFCFG_FCFGR:SWFRST); a software triggered soft reset; or a hard reset

10.4 Starting the EEPROM Emulation Flash memory automatic algorithm

Writing to and erasing Flash memory is performed by launching the Flash memory's own automatic algorithms. The read/reset, write, macro erase and sector erase are available. The erase suspend and resume function is available during sector erase.

Command sequence table

With reference to Table 10-18, automatic algorithms for Flash memory write or erase are launched by writing one to six bytes or half-words or words to the Flash memory in succession

The command data must be written to the lower byte (except the Program Data (PD) of the write command). The data written to the upper bytes (in case of half-word or word access) is ignored.

The data width used for the fourth bus write cycle of the write command (PA and PD) determines the write mode of the Flash (byte or half-word or word).

Table 10-18. Command sequence table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/reset	1	mmmm XXXX	F0	-	-	-	-	-	-	-	-	-	-
Write/ program	4	mmmm XAA8	AA	mmmm X554	55	mmmm XAA8	A0	PA	PD	-	-	-	-
Macro erase	6	mmmm XAA8	AA	mmmm X554	55	mmmm XAA8	80	mmmm XAA8	AA	mmmm X554	55	mmmm XAA8	10
Sector erase	6	mmmm XAA8	AA	mmmm X554	55	mmmm XAA8	80	mmmm XAA8	AA	mmmm X554	55	SA	30
Sector erase suspend	1	mmmm XXXX	B0	-	-	-	-	-	-	-	-	-	-
Sector erase resume	1	mmmm XXXX	30	-	-	-	-	-	-	-	-	-	-
Unlock bypass set	3	mmmm XAA8	AA	mmmm X554	55	mmmm XAA8	20	-	-	-	-	-	-
Unlock bypass program (*1.)	2	mmmm XXXX	A0	PA	PD	-	-	-	-	-	-	-	-
Unlock bypass reset (*1.)	2	mmmm XXXX	90	mmmm XXXX	XX	-	-	-	-	-	-	-	-

mmmm: Flash memory address, X: Arbitrary value (0/1), PA: Program Address, PD: Program Data, SA: Sector Address

Notes:

- The addresses in the table are the values in the CPU memory map. All addresses and data are represented using hexadecimal notation
- CPU/Flash data bits [31:8] are ignored for command writing (except for the Program Data (PD))
- In the event of a fourth write cycle in the write/program sequence, the PA should be the actual write/PA location and PD the data*2 to be written
- PA should be aligned*3 to a 4 byte grid (for a 32-bit write)
- The ECC is written according to the configuration of the EEFCFG_WMER:MME and EEFCFG_WMER:AME bits

- Similarly during the 6th write cycle of the sector erase command sequence, the Sector Address is the actual address of where the sector erase command is meant to operate. Flash macro selection and sector selection is done in the same way as for the PA
- Writing an illegal address or data, or writing them in the incorrect order resets the Flash memory to read mode
- It is possible to read data from the Flash between the command write cycles. The command execution starts after the last write cycle
- Commands must not be written to sectors which are write protected. Any write access to a write protected sector activates the hardware reset of the corresponding Flash module and forces the corresponding RDY bit to 0 for 20 μs. Such a hardware reset initializes the Flash to the read/ reset state, independent of the previous state (program, erase, suspend)

1. This command is valid only in unlock bypass mode.
2. The PD should be 32-bit data if ECC is enabled. PD can be 8-, 16- or 32-bit wide if ECC is disabled.
3. The PA should be aligned to a 2-byte grid for half-word access and a 4-byte grid for word access.

Note:

If half-word and byte write is used, the ECC bits are not programmed. Thus it is recommended to use this only when ECC is disabled to avoid errors at read back.

Flash parallel programming

- Flash parallel programming means direct Flash addressing (with no address translation required) with the upper bits used for Flash macro selection
- Thus in the event of Flash parallel programming, ’mmmm’ in the table can point to any of the Flash address ranges shown in Figure 10-20
- In the event of a fourth write cycle in the write/program sequence, the PA is the write address. Only even addresses can be specified in half-word mode. For byte mode both even and odd addresses are permitted. PD, i.e. write data, should be 8 or 16 bits wide
- Also ECC pins are directly mapped to external pins in this mode and ECC calculation and checking should be handled by the Flash programmer tool

10.5 Notes on using Flash memory

This section contains notes on using Flash Memory. For more details on Flash Memory, refer to 9. Tightly Coupled Flash and Confirming the automatic algorithm execution state on page 526 and Writing to and erasing Flash memory on page 533.

Input of a CPU reset

A CPU reset (power reset, external reset, software reset, watchdog reset or clock stop reset, software triggered hardware reset, clock supervisor reset, PD3 reset) asserts the Flash hardware reset. A reset assertion during Flash writing causes the data written to be undefined.

Resetting the device once execution of a sector erase has begun corrupts the data in the sector. In that case, restart the erase on this sector and allow it to complete. If a reset occurred during writing, rewrite with the same data.

Register configuration

Since a Flash reset is asserted for a minimum period of 500 ns during a software triggered Flash reset, the software should wait until the Flash reset has completed before starting memory access. To know that the reset has completed and the Flash is ready for the next command, the software can read the EEFCFG_SR:RDY bit (the RDY output from Flash macro returns to ’1’ after a time period that does not exceed 80 μs).

In addition, the CAM output from the Flash macro is not valid during reset assertion and for 70 ns after reset release. Thus it should be ensured that the TCFCFG_FCAMLRn and TCFCFG_FCAMHRn registers (in the Tightly Coupled Flash) are not accessed during this period.

Program access to Flash memory

While the Flash memory is being erased or data written to it, reading is not possible. Hence, when it is required to read data from the Flash memory while the memory is being erased or written to, the required data must first be copied to RAM before starting the erase/write operation. A sector erase operation can be suspended, during which time it is possible to read data from other sectors. A write command during the suspend state is ignored while a resume command is not.

Note: After restarting a sector erase, a minimum wait time of 2 ms must be applied before submitting a ‘sector erase suspend’ command.

Flash programming and hardware sequence flags

- The status of the Flash can be monitored by reading the hardware sequence flags
- Since hardware sequence flags are not ECC protected, the ECC check is disabled when the Flash auto-algorithm is in progress. Also in case of a 32-bit write
- An interrupt generated based on EEFCFG_SR:RDYINT = ’1’ and EEFCFG_ICR:RDYIE = ’1’ should be used to know the auto-algorithm status for a read protected sector

Flash hardware reset

Configuring EEFCFG_CR:SWFRST to ’1’ asserts the Flash macro reset. The software should check the status of the Flash by reading EEFCFG_SR:RDY. Until EEFCFG_SR:RDY equals ‘1’, the software should not issue the next access to Flash memory.

10.6 Notes on enabling ECC

This section illustrates the manual and automatic enabling of ECC when ECC is ON and access is via EEFLASH_ECC_MIR.

Figure 10-21. Writing data in manual mode

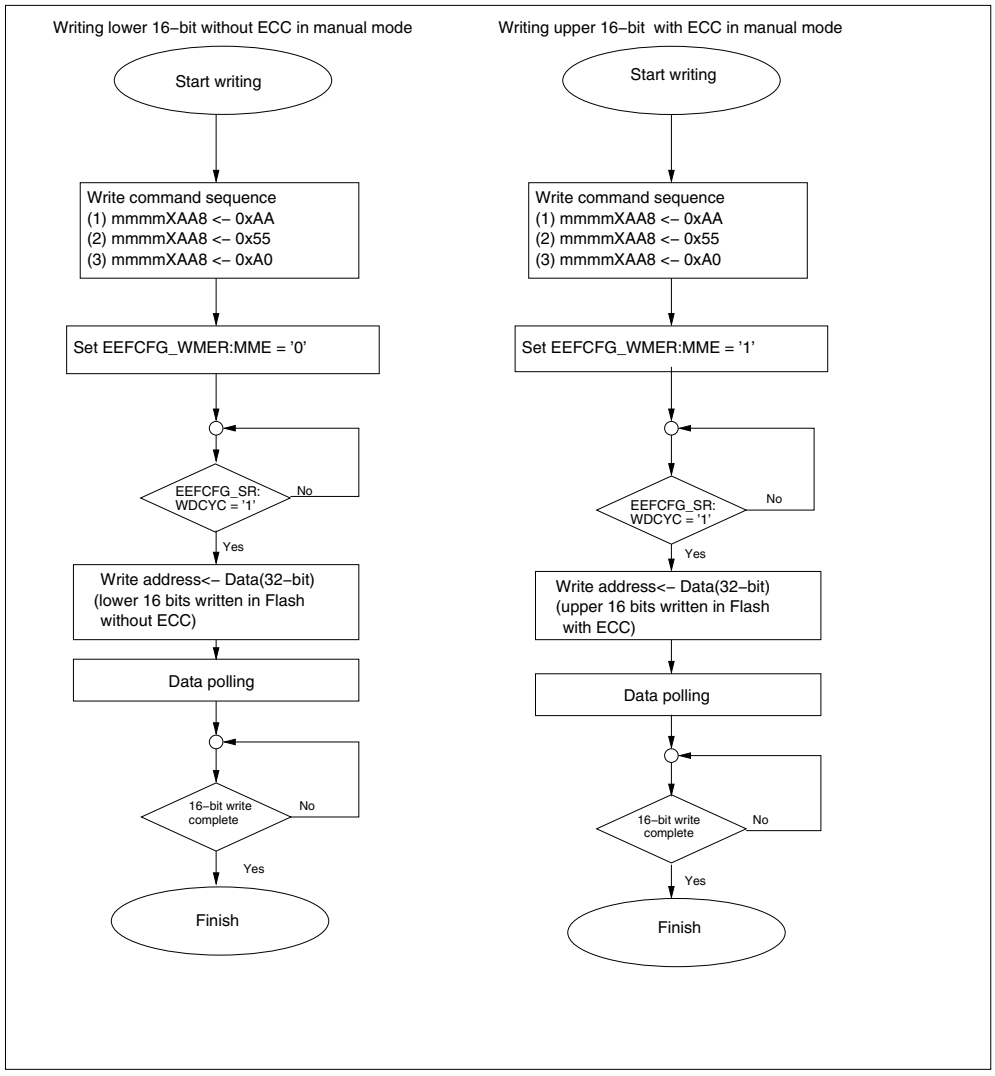


Figure 10-22. Writing data in automatic mode

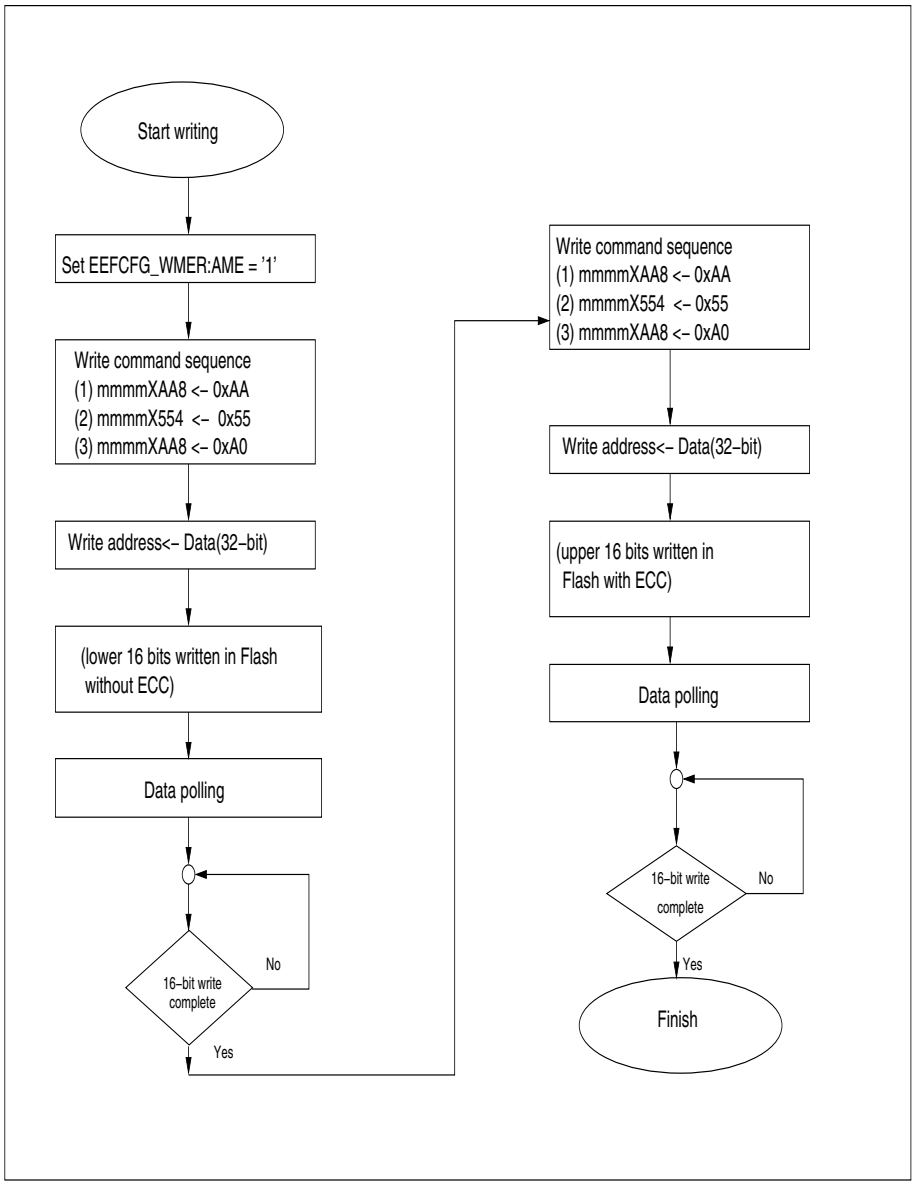
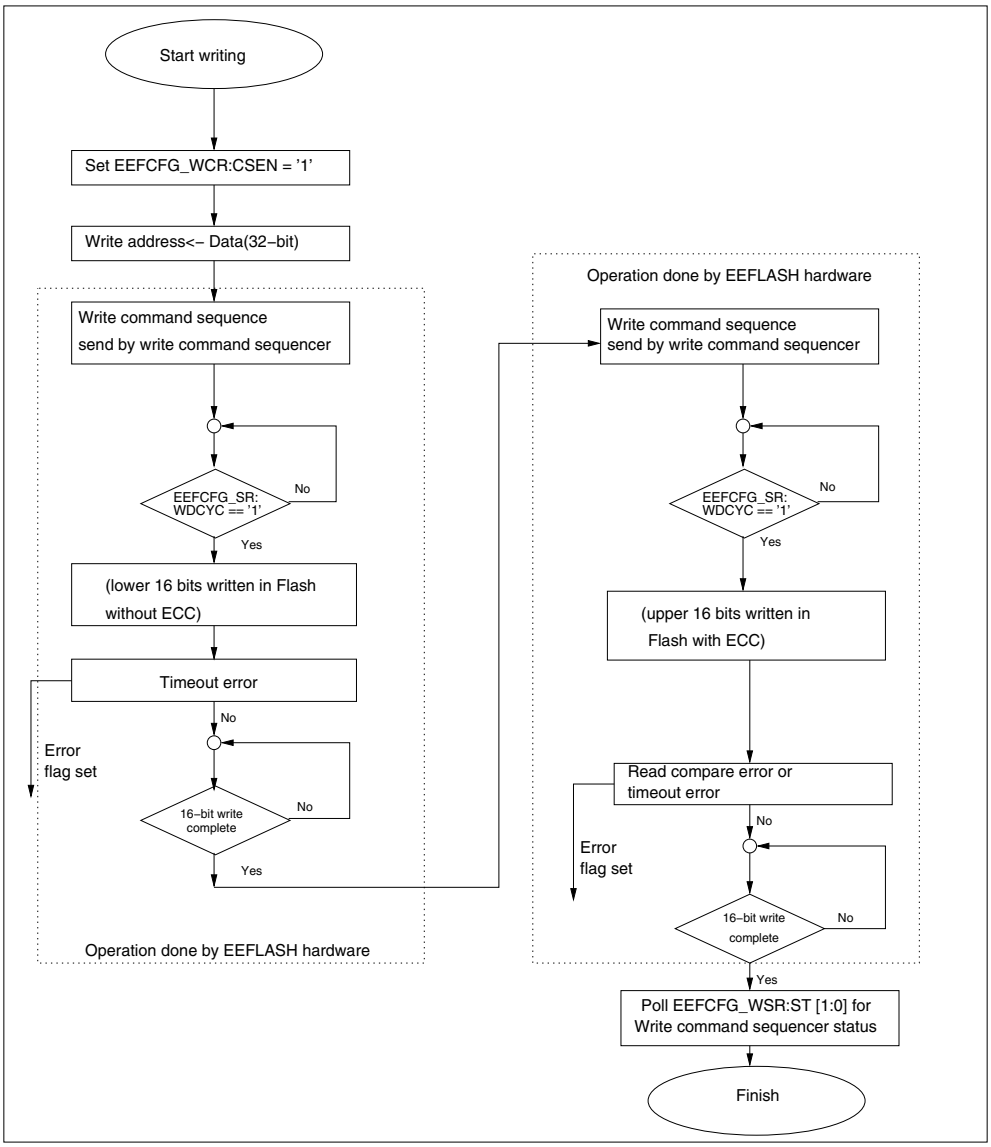


Figure 10-23. Writing data in write command sequencer mode



Note: In write command sequencer mode, sending an automatic write command to Flash and enabling the ECC is taken care by the EEPROM Emulation Interface. The user only needs to send the write data.

11. EEPROM Emulation Flash and SHE

This chapter explains the function and operation of the EEPROM Emulation Flash module and is applicable to devices that are equipped with the Secure Hardware Extension (SHE).

11.1 Outline of the EEPROM Emulation Flash

This section describes the features and the block diagram of the EEPROM Emulation Flash module.

Features of EEPROM Emulation Flash

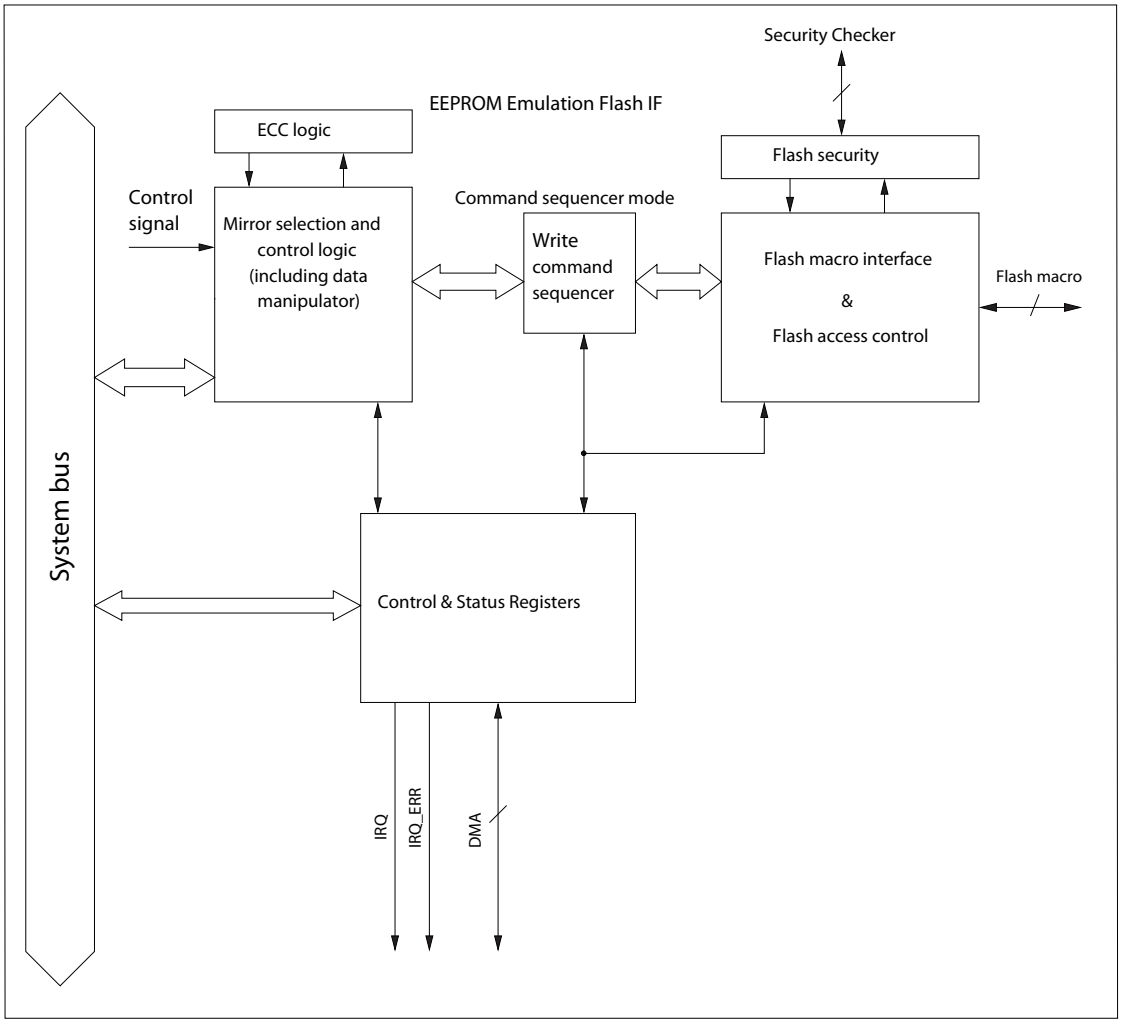
The EEPROM Emulation Flash memory is used to store data. EEPROM Emulation Flash write or erase can be done using instructions from the Central Processing Unit (CPU) or any other master via the EEPROM Emulation Flash Interface.

Other features of the EEPROM Emulation Flash are:

- It supports Flash read by the CPU in 8-,16- and 32-bit wide
- Flash write by the CPU is restricted to only 32-bit access independent of the ECC
- Flash programming can be done in CPU mode using the command sequencer
- It supports data programming by Direct Memory Access (DMA)
- It supports an ECC scheme for Single-Bit Error Correction and Double-Bit Error Detection (SEC-DED)
- ECC logic can be switched off using a configuration bit
- It supports ECC testing by injecting errors
- Write access to the Flash Configuration Register settings are key protected
- It supports sector-wise read and write access permissions
- Flash write or erase is possible in MCU privileged mode and non-privileged mode
- Access to a write protected sector results in a bus error response
- Access to a read protected sector results in a bus error response
- Access to an unused address range of Flash returns undefined data and generates a bus error response
- It detects the completion of write or erase commands using CPU interrupts

Block diagram of EEPROM Emulation Flash Interface

Figure 11-1. Block diagram of EEPROM Emulation Flash Interface



11.2 EEPROM Emulation Flash Memory Interface registers

This section describes the registers of the EEPROM Emulation Flash Memory Interface.

Registers of the EEPROM Emulation Flash Memory Interface

The following registers are available for the EEPROM Emulation Flash:

- Configuration Protection Key Register (EEFCFG_CPR)
- Configuration Register (EEFCFG_CR)
- ECC Control Register (EEFCFG_ECR)
- Write Command Sequencer Configuration Register (EEFCFG_WCR)
- Write Command Sequencer Status Register (EEFCFG_WSR)
- Data Bit Error Injection Register (EEFCFG_DBEIR)
- ECC Bit Error Injection Register (EEFCFG_EEIR)
- Interrupt Control Register (EEFCFG_ICR)
- Status Register (EEFCFG_SR)
- SEC Interrupt Register (EEFCFG_SECIR)
- ECC Error Address Register (EEFCFG_EEAR)
- Module Identification Register (EEFCFG_MIR)
- Extra Mode Enable Register (EEFCFG_EMENR)
- Flash Register (EEFCFG_FCAMLR)
- Flash Register (EEFCFG_FCAMHR)
- Sequencer Write Mode Register (EEFCFG_SEQWM)
- Sequencer Command Mode Register (EEFCFG_SEQCM)
- Arbitration Error Register (EEFCFG_ARBERR)
- Read Arbitration Error Register (EEFCFG_ARBCLR)
- Bus Error Response Register (EEFCFG_BERR)
- Bus Error Response Clear Register (EEFCFG_BERRCLR)

Registers with changed functionality

Compared to the registers described in the previous chapter, the functionality of the following registers has changed in the EEFLASH Emulation Flash with SHE support:

- EEFCFG_CR - EEFCFG_CR:FAWC prohibits configuration of 0, 1 or 3 wait cycles
- EEFCFG_WCR - EEFCFG_WCR:CSEN is always ‘1’ - the command sequencer can’t be disabled
- EEFCFG_WSR - EEFCFG_WSR:ST never returns ‘11’ - the command sequencer can’t be disabled

Removed registers

The EEFCFG_WMER register does not exist in an EEFLASH Emulation Flash with SHE support because the automatic algorithm cannot be triggered manually. New functionality supported by the command sequencer is covered by the Sequencer Write Mode Register (EEFCFG_SEQWM).

Memory layout of EEPROM Emulation Flash Memory Interface registers

Table 11-1. Memory layout of EEPROM Emulation Flash Memory Interface registers

Offset	+3	+2	+1	+0
0x00000000	EEFCFG_CPR 00000000 00000000 00000000 00000000			

Table 11-1. Memory layout of EEPROM Emulation Flash Memory Interface registers

Offset	+3	+2	+1	+0
0x00000004	reserved 00000000 00000000 00000000 00000000			
0x00000008	EEFCFG_CR 00000000 00000000 00000000 00000010			
0x0000000C	EEFCFG_ECR 00000000 00000000 00000000 00000000			
0x00000010	EEFCFG_WCR 00000000 00000000 00000000 00000000			
0x00000014	EEFCFG_WSR 00000000 00000000 00000000 00000000			
0x00000018	EEFCFG_DBEIR 00000000 00000000 00000000 00000000			
0x0000001C	EEFCFG_EEIR 00000000 00000000 00000000 00000000			
0x00000020	reserved 00000000 00000000 00000000 00000000			
0x00000024	EEFCFG_ICR 00000000 00000000 00000000 00000000			
0x00000028	EEFCFG_SR 00000000 00000000 00000000 00000000			
0x0000002C	EEFCFG_SECIR 00000000 00000000 00000000 00000000			
0x00000030	EEFCFG_EEAR 00000000 00000000 00000000 00000000			
0x00000034	EEFCFG_MIR 00000000 00000000 00000000 00000000			
0x00000038	EEFCFG_EMENR 00000000 00000000 00000001 00000000			
0x0000003C	reserved 00000000 00000000 00000000 00000000			
0x00000040	reserved 00000000 00000000 00000000 00000000			
0x00000044	reserved 00000000 00000000 00000000 00000000			
0x00000048	EEFCFG_FCAMLR XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000004C	EEFCFG_FCAMHR 00000000 00000000 000XXXXX XXXXXXXX			

Table 11-1. Memory layout of EEPROM Emulation Flash Memory Interface registers

Offset	+3	+2	+1	+0
0x00000050	EEFCFG_SEQWM 00000000 00000000 00000000 00000000			
0x00000054	EEFCFG_SEQCM 00000000 00000000 00000000 00000000			
0x00000058	EEFCFG_ARBERR 00000000 00000000 00000000 00000000			
0x0000005C	EEFCFG_ARBCLR 00000000 00000000 00000000 00000000			
0x00000060	EEFCFG_BERR 00000000 00000000 00000000 00000000			
0x00000064	EEFCFG_BERRCLR 00000000 00000000 00000000 00000000			

11.2.1 Configuration Protection Key Register (EEFCFG_CPR)

The Configuration Protection Key Register (EEFCFG_CPR) should be written with the correct key value (0xCF6DF1A5) to unlock write access to the Interface Configuration Register (EEFCFG_CR) or the ECC Control Register (EEFCFG_ECR) and the ECC Bit Error Injection Registers (EEFCFG_DBEIR & EEFCFG_EEIR). Access is locked again after the next write on the Advanced High-Performance Configuration Bus (AHB). An incorrect key value or writing to the EEPROM Emulation Flash configuration registers without unlocking or any deviation in the above protection sequence causes an error response on the slave bus and hence a data abort. This register must be written using 32-bit writes only.

Configuration Protection Key Register (EEFCFG_CPR)

Figure 11-2. Configuration Protection Key Register (EEFCFG_CPR)

EEFCFG_CPR																															
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**
CPR[31]	CPR[30]	CPR[29]	CPR[28]	CPR[27]	CPR[26]	CPR[25]	CPR[24]	CPR[23]	CPR[22]	CPR[21]	CPR[20]	CPR[19]	CPR[18]	CPR[17]	CPR[16]	CPR[15]	CPR[14]	CPR[13]	CPR[12]	CPR[11]	CPR[10]	CPR[9]	CPR[8]	CPR[7]	CPR[6]	CPR[5]	CPR[4]	CPR[3]	CPR[2]	CPR[1]	CPR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-2. Configuration Protection Key Register (EEFCFG_CPR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CPR	Configuration Protection Register This register protects the EEPROM Emulation Flash Configuration Register (EEFCFG_CR), the EEPROM Emulation Flash ECC Control Register (EEFCFG_ECR) and the EEPROM Emulation Flash ECC Error Injection Registers (EEFCFG_DBEIR and EEFCFG_EEIR) from being accidentally modified by software. Writing the correct key value (i.e. 0xCF6DF1A5) to this register unlocks write access to the EEFCFG_CR, EEFCFG_ECR, EEFCFG_DBEIR and EEFCFG_EEIR registers. Writing any other value to this register causes a data abort. Reading this register always returns 0xFFFFFFFF when write access to EEFCFG_CR, EEFCFG_ECR, EEFCFG_DBEIR and EEFCFG_EEIR is unlocked. Reading this register always returns 0x00000000 when write access to EEFCFG_CR, EEFCFG_ECR, EEFCFG_DBEIR, and EEFCFG_EEIR is locked (default).

11.2.2 Configuration Register (EEFCFG_CR)

The Configuration Register (EEFCFG_CR) controls wait-cycle settings for EEPROM Emulation Flash access. This register also contains the EEPROM Emulation Flash write access enable and the EEPROM Emulation Flash reset control bit. This register is writable only if access is unlocked by writing the correct key value to the EEFCFG_CPR register.

Configuration Register (EEFCFG_CR)

Figure 11-3. Configuration Register (EEFCFG_CR)

EEFCFG_CR																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SWFRST	read0	read0	read0	read0	read0	read0	read0	WE	read0	read0	read0	read0	read0	read0	FAWC[1]	FAWC[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WPS1	R0	R0	R0	R0	R0	R0	R0	RWPS	R0	R0	R0	R0	R0	R0	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 11-3. Configuration Register (EEFCFG_CR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	SWFRST	Software Triggered Flash Reset This bit is used to reset the Flash macro by software. It also resets the write command sequencer if it is enabled. '0': No effect '1': Trigger Flash macro and write command sequencer reset The time to reset the Flash macro is 500 ns. This time is counted with respect to the maximum Flash operating frequency. Therefore, for slow frequencies it will always be more than required. Reading this bit returns '0'.
[15:9]	read0	-
[8]	WE	Flash Write Enable This bit enables/disables writing to the Flash. '0': Flash write is disabled (default) '1': Flash write is enabled Write access to Flash with this bit set to '0' results in a bus error response.
[7:2]	read0	-

Table 11-3. Configuration Register (EEFCFG_CR) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	FAWC	Flash Access Wait Cycle Control These bits define the number of wait cycles required for Flash read and write access. These bits should be set based on the system clock frequency and Flash access time. Read/write wait cycles. '00': 0 wait cycles '01': 1 wait cycle '10': 2 wait cycles (default) '11': 3 wait cycles The relationship between the wait cycle setting and clock frequency is given as: ■ FAWC = (system clock frequency/Flash operating frequency) - 1 For SHE equipped devices only 2 wait cycles can be configured for security reasons

11.2.3 ECC Control Register (EEFCFG_ECR)

The ECC Control Register (EEFCFG_ECR) controls the disabling of the ECC logic. This register is writable only if access is unlocked by writing the correct key value to the EEFCFG_CPR register. This register is writable to only once; writing more than once leads to a bus error response.

ECC Control Register (EEFCFG_ECR)

Figure 11-4. ECC Control Register (EEFCFG_ECR)

EEFCFG_ECR																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ECCOFF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-4. ECC Control Register (EEFCFG_ECR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ECCOFF	ECC Disable/Switch-Off This bit disables ECC generation and checking. '0': ECC generation/checking is ON (default) 1': ECC generation/checking is OFF **Note:** ECC generation and checking are disabled irrespective of the used mirror scheme. Therefore when EEFCFG_ECR:ECCOFF is disabled, ECC generation and checking are disabled when writing to or reading from any mirror.

11.2.4 Write Command Sequencer Configuration Register (EEFCFG_WCR)

The Write Command Sequencer Configuration Register (EEFCFG_WCR) is used to configure the command sequencer and DMA.

Write Command Sequencer Configuration Register (EEFCFG_WCR)

Figure 11-5. Write Command Sequencer Configuration Register (EEFCFG_WCR)

EEFCFG_WCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DMAEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-5. Write Command Sequencer Configuration Register (EEFCFG_WCR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	DMAEN	DMA Mode Enable '0': Disables DMA mode 1': Enables DMA mode The EEPROM Emulation Flash Interface generates a DMA request when EEFCFG_WCR:DMAEN is set and EEFCFG_WSR:ST is '00'.

11.2.5 Write Command Sequencer Status Register (EEFCFG_WSR)

The Write Command Sequencer Status Register (EEFCFG_WSR) returns the status of the command sequencer.

Write Command Sequencer Status Register (EEFCFG_WSR)

Figure 11-6. Write Command Sequencer Status Register (EEFCFG_WSR)

EEFCFG_WSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ST[1]	ST[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-6. Write Command Sequencer Status Register (EEFCFG_WSR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1:0]	ST	Write Command Sequencer Status bits '00': The write command sequencer is in idle state. In idle state, the sequencer is ready to receive a new write command '01': The write command sequencer submits the write command to the Flash module '10': The write command sequencer checks the progress of the write command by checking the Flash RDY status and the Flash hardware sequence flags '11': Reserved Do not access the Flash (read or write) as long as the command sequencer is not in idle state.

11.2.6 Data Bit Error Injection Register (EEFCFG_DBEIR)

Data Bit Error Injection Register (EEFCFG_DBEIR) is used to inject errors into the data part of the EEPROM Emulation Flash read data to test the ECC logic. It is writable only if access is unlocked by writing the correct key value to the EEFCFG_CPR register.

Data Bit Error Injection Register (EEFCFG_DBEIR)

Figure 11-7. Data Bit Error Injection Register (EEFCFG_DBEIR)

EEFCFG_DBEIR																															
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**
DBEIR[31]	DBEIR[30]	DBEIR[29]	DBEIR[28]	DBEIR[27]	DBEIR[26]	DBEIR[25]	DBEIR[24]	DBEIR[23]	DBEIR[22]	DBEIR[21]	DBEIR[20]	DBEIR[19]	DBEIR[18]	DBEIR[17]	DBEIR[16]	DBEIR[15]	DBEIR[14]	DBEIR[13]	DBEIR[12]	DBEIR[11]	DBEIR[10]	DBEIR[9]	DBEIR[8]	DBEIR[7]	DBEIR[6]	DBEIR[5]	DBEIR[4]	DBEIR[3]	DBEIR[2]	DBEIR[1]	DBEIR[0]
RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-7. Data Bit Error Injection Register (EEFCFG_DBEIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DBEIR	Data Bit Error Injection This bit is used to inject errors into data bits of Flash read data to test the ECC logic. Error injection is done by inverting respective data bits of the Flash read data before passing it to the ECC checking logic. '0': No effect on the corresponding bit of the Flash read data (default) '1': Flips the corresponding bit of the Flash read data Errors are injected independent from EEFCFG_ECR:ECCOFF and the accessed mirror

11.2.7 ECC Bit Error Injection Register (EEFCFG_EEIR)

The ECC Bit Error Injection Register (EEFCFG_EEIR) is used to inject errors into ECC bits of the EEPROM Emulation Flash read data to test the ECC logic. It is writable to only if access is unlocked by writing the correct key value to the EEFCFG_CPR register.

ECC Bit Error Injection Register (EEFCFG_EEIR)

Figure 11-8. ECC Bit Error Injection Register (EEFCFG_EEIR)

EEFCFG_EEIR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	EEIR[6]	EEIR[5]	EEIR[4]	EEIR[3]	EEIR[2]	EEIR[1]	EEIR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS	RWPS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-8. ECC Bit Error Injection Register (EEFCFG_EEIR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6:0]	EEIR	ECC Bit Error Injection This bit is used to inject errors into the ECC bits of Flash read data to test the ECC logic. Error injection is done by inverting the respective ECC bits of Flash read data before passing it to the ECC checking logic. '0': No effect on the corresponding ECC bit of the Flash read data (default) '1': Flips the corresponding ECC bit of the Flash read data Errors are injected independent from EEFCFG_ECR:ECCOFF and the accessed mirror

11.2.8 Interrupt Control Register (EEFCFG_ICR)

The Interrupt Control Register (EEFCFG_ICR) contains the interrupt enable and clear bits.

Interrupt Control Register (EEFCFG_ICR)

Figure 11-9. Interrupt Control Register (EEFCFG_ICR)

EEFCFG_ICR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	HANGIC	RDYIC	read0	read0	read0	read0	read0	read0	HANGIE	RDYIE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0WP1	R0	R0	R0	R0	R0	R0	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-9. Interrupt Control Register (EEFCFG_ICR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	HANGIC	Hang Interrupt Clear '0': No effect '1': Clears EEFCFG_SR:HANGINT bit (Hang Interrupt Status bit) Reading this bit returns '0'.
[8]	RDYIC	Flash Ready Interrupt Clear '0': No effect '1': Clears EEFCFG_SR:RDYINT bit (Flash Ready Interrupt Status bit) Reading this bit returns '0'.
[7:2]	read0	-
[1]	HANGIE	Hang Interrupt Enable '0': Disables hang interrupt on error interrupt line (default) '1': Enables hang interrupt on error interrupt line
[0]	RDYIE	Flash Ready Interrupt Enable '0': Disables Flash ready interrupt (default) '1': Enables Flash ready interrupt

11.2.9 Status Register (EEFCFG_SR)

The Status Register (EEFCFG_SR) holds the status of the RDY EEPROM Emulation Flash memory output and its respective interrupt status.

Status Register (EEFCFG_SR)

Figure 11-10. Status Register (EEFCFG_SR)

EEFCFG_SR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	HANGINT	RDYINT	read0	read0	read0	read0	read0	read0	read0	RDY
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-10. Status Register (EEFCFG_SR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	HANGINT	Hang Interrupt This bit is set at the rising edge of the Flash HANG output. It is cleared by writing '1' to EEFCFG_ICR:HANGIC (Hang Interrupt Clear bit). '0': Hang condition has not occurred (default) '1': Hang condition has occurred This is a read-only bit; writing to this bit has no effect.
[8]	RDYINT	Flash Ready Interrupt This bit is set on the rising edge of the Flash RDY output. It is cleared by writing '1' to EEFCFG_ICR:RDYIC (RDY Interrupt Clear bit). '0': The rising edge of the Flash RDY output is not detected (default) '1': The rising edge of the Flash RDY output is detected (i.e. Flash write or erase operation is completed) This is a read-only bit; writing to this bit has no effect.
[7:1]	read0	-

Table 11-10. Status Register (EEFCFG_SR) bits

Bit Position	Bit Field Name	Bit Description
[0]	RDY	Flash Ready Status This bit reflects the status (sampled) of the Flash RDY output. In other words, it indicates whether or not the Flash macro is ready to accept a new command. '0': Flash programming is in progress; only a read/reset or suspend command is accepted '1': Flash is returned from auto-algorithm (program/erase) and is ready for a new command (default) This is a read-only bit; writing to it has no effect.

11.2.10 SEC Interrupt Register (EEFCFG_SECIR)

The SEC Interrupt Register (EEFCFG_SECIR) contains the ECC single-bit error interrupt status and corresponding interrupt enable and clear bits.

SEC Interrupt Register (EEFCFG_SECIR)

Figure 11-11. SEC Interrupt Register (EEFCFG_SECIR)

EEFCFG_SECIR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SECINT	read0	read0	read0	read0	read0	read0	read0	SECIC	read0	read0	read0	read0	read0	read0	read0	SECIE
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-11. SEC Interrupt Register (EEFCFG_SECIR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	SECINT	ECC Single Error Correction Interrupt This bit is set when a single-bit error is detected/corrected during ECC checking. It is cleared by writing '1' to EEFCFG_SECIR:SECIC (ECC Single Error Interrupt Clear bit). '0': ECC single error has not occurred (default) '1': ECC single error has occurred This is a read-only bit, writing to this bit has no effect. Setting of EEFCFG_SECIR:SECINT has higher priority than clearing it.
[15:9]	read0	-
[8]	SECIC	ECC Single Error Correction Interrupt Clear '0': No effect '1': Clears EEFCFG_SECIR:SECINT (ECC Single Error Interrupt bit) Reading this bit returns '0'.
[7:1]	read0	-

Table 11-11. SEC Interrupt Register (EEFCFG_SECIR) bits

Bit Position	Bit Field Name	Bit Description
[0]	SECIE	ECC Single Error Correction Interrupt Enable '0': Disables ECC Single Error Correction interrupt (default) '1': Enables ECC Single Error Correction interrupt

11.2.11 ECC Error Address Register (EEFCFG_EEAR)

The ECC Error Address Register (EEFCFG_EEAR) stores the address of the EEPROM Emulation Flash memory location where the most recent ECC single bit error has occurred. Software can use this register to find out the error location.

ECC Error Address Register (EEFCFG_EEAR)

Figure 11-12. ECC Error Address Register (EEFCFG_EEAR)

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

Table 11-12. ECC Error Address Register (EEFCFG_EEAR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EEAR	ECC Error Address Register These read-only bits contain the 32-bit aligned Flash address of where the most recent ECC single bit error has occurred. This address may be helpful in marking the memory location as a weak cell. Writing to these bits has no effect.

11.2.12 Module Identification Register (EEFCFG_MIR)

The Module Identification Register (EEFCFG_MIR) contains the Module Identification Number, revision and patch details of the EEPROM Emulation Flash Interface module.

Module Identification Register (EEFCFG_MIR)

Figure 11-13. Module Identification Register (EEFCFG_MIR)

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

Table 11-13. Module Identification Register (EEFCFG_MIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module Identification Number This register identifies the particular version of the hardware used in the device. The information contained in the register helps in the development of software for the particular hardware version. **Note:** For more details on the module ID number, refer to the device-specific datasheet.

11.2.13 Extra Mode Enable Register (EEFCFG_EMENR)

The Extra Mode Enable Register (EEFCFG_EMENR) contains the extra mode control bit.

Extra Mode Enable Register (EEFCFG_EMENR)

Figure 11-14. Extra Mode Enable Register (EEFCFG_EMENR)

EEFCFG_EMENR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AEE	read0	read0	read0	read0	read0	read0	read0	EMEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp	R0	R0	R0	R0	R0	R0	R0	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 11-14. Extra Mode Enable Register (EEFCFG_EMENR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	AEE	Arbitration Error Enable The default value for this bit is '1' (enabled). This bit can be disabled during wait states in which all buses may stall. In the worst case, a watchdog reset may occur.
[7:1]	read0	-
[0]	EMEN	Extra Mode Enable The software should always write this bit to '0'. Writing '1' leads to unknown behavior.

11.2.14 Flash Register (EEFCFG_FCAMLR)

The Flash CAM output lower register (EEFCFG_FCAMLR) stores bits[31:0] of the Flash macro CAM output.

Flash Register (EEFCFG_FCAMLR)

Figure 11-15. Flash CAM output lower register (EEFCFG_FCAMLR)

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

Table 11-15. Flash CAM output lower register (EEFCFG_FCAMLR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	CAML	Flash CAM Output Lower Word This is a read-only register. Reading this register returns the lower word of the Flash macro CAM output.

11.2.15 Flash Register (EEFCFG_FCAMHR)

The Flash CAM output upper register (EEFCFG_FCAMHR) stores bits[44:32] of the Flash macro CAM.

Flash Register (EEFCFG_FCAMHR)

Figure 11-16. Flash CAM output upper register (EEFCFG_FCAMHR)

EEFCFG_FCAMHR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CAMH[12]	CAMH[11]	CAMH[10]	CAMH[9]	CAMH[8]	CAMH[7]	CAMH[6]	CAMH[5]	CAMH[4]	CAMH[3]	CAMH[2]	CAMH[1]	CAMH[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 11-16. Flash CAM output upper register (EEFCFG_FCAMHR)

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:13]	read0	-
[12:0]	CAMH	Flash CAM Output Upper Word This is a read-only register. Reading this register returns the upper word of the Flash macro CAM output.

11.2.16 Sequencer Write Mode Register (EEFCFG_SEQWM)

The Sequencer Write Mode Register (EEFCFG_SEQWM) selects between 16 and 32 bit write mode.

Sequencer Write Mode register EEFCFG_SEQWM

Figure 11-17. Sequencer write mode register EEFCFG_SEQWM

EEFCFG_SEQWM																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WM16
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-17. Sequencer Command register EECFG_SEQWM bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	WM16	Write mode selection '0': 32-bit write mode enabled '1': 16-bit write mode enabled

Note: Data must be written with 32-bit accesses independent of the EEFCFG_SEQWM:WM16 bit.

16-bit write mode:

Writing 32-bit data to the EEFLASH_NOECC_MIR results in the programming of the lower 16 bits of data to the Flash memory without ECC parity bits. Writing 32-bit data to the EEFLASH_ECC_MIR results in writing the upper 16-bit of data and, if EEFCFG_ECR:ECCOFF is not set, the ECC parity bits calculated from the 32-bit write data.

32-bit write mode:

Writing 32-bit data to EEFLASH_NOECC_MIR results in the programming of 32-bit data without ECC parity bits. Writing 32-bit data to EEFLASH_ECC_MIR results in the programming of 32-bit data, including ECC parity bits.

11.2.17 Sequencer Command Mode Register (EEFCFG_SEQCM)

The Sequencer Command Mode Register (EEFCFG_SEQCM) allows the triggering of the read reset and sector erase commands.

Sequencer Command Mode register EEFCFG_SEQCM

Figure 11-18. Sequencer command mode register EEFCFG_SEQCM

EEFCFG_SEQCM																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	ERS7E	ERS6E	ERS5E	ERS4E	ERS3E	ERS2E	ERS1E	ERS0E	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	OPC[1]	OPC[0]
R0	R0	R0	R0	R0	R0	R0	R0	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWp	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-18. Sequencer Command mode register EEFCFG_SEQCM bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	ERS7E	Erase Sector 7 Write '1' to erase sector 7 **Note:** Sector 7 is only available on devices without SHE
[22]	ERS6E	Erase Sector 6 Write '1' to erase sector 6 **Note:** Sector 6 is only available on devices without SHE
[21]	ERS5E	Erase Sector 5 Write '1' to erase sector 5
[20]	ERS4E	Erase Sector 4 Write '1' to erase sector 4
[19]	ERS3E	Erase Sector 3 Write '1' to erase sector 3
[18]	ERS2E	Erase Sector 2 Write '1' to erase sector 2
[17]	ERS1E	Erase Sector 1 Write '1' to erase sector 1
[16]	ERS0E	Erase Sector 0 Write '1' to erase sector 0

Table 11-18. Sequencer Command mode register EEFCFG_SEQCM bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:2]	read0	-
[1:0]	OPC	OPC[1:0] selects a command: '00': Idle '01': Perform read reset command '10': Perform sector erase command '11': Reserved

Notes:

1. A new command trigger shall only be written if the two OPC bits indicate an idle state and EEFCFG_SR:RDY is high.
2. If the OPC bits indicate a read/reset command, the ERS*E bits must contain a valid sector selection, even if it is ‘don’t care’ for this command.
3. If the OPC bits indicate a sector erase command, then the command is only executed if:
   - At least one sector is selected in the ERS*E bits and
   - None of the selected sectors is write protected
4. If the sector erase command has not been executed because the above two conditions have not been fulfilled, then the OPC bits indicate an idle state after writing the sector erase command.
5. If a write protected sector is selected for erase, then a bus error response is returned.
6. The OPC and ERS*E bits can be written at once or independent of each other, e.g. using byte or halfword accesses.
7. If multiple sectors are selected for erasing, then the sectors are erased in descending order.
8. If a command is accepted, then EEFCFG_SR:RDY is low and the OPC bits indicate the written command one clock cycle after writing OPC.
9. The command execution is finished if EEFCFG_SR:RDY returns ‘1’ on read. The OPC bits indicate an idle state again and the ERS*E bits are all ‘0’.
10. The execution time of the commands depends on the arbitration state between the public and private interface because an ongoing command execution on the opposite interface may have to finish first.
11. The execution of a command does not depend on the state of the write enable bit EEFCFG_CR:WE which is valid for data writing only.
12. Writing a reserved opcode is ignored, i.e. the OPC bits indicate an idle state after writing the sector erase command.

11.2.18 Arbitration Error Register (EEFCFG_ARBERR)

The Arbitration Error Register (EEFCFG_ARBERR) indicates a read was requested when the SHE side owned the arbitration.

Read arbitration Error register EEFCFG_ARBERR

Figure 11-19. Read arbitration Error register EEFCFG_ARBERR

EEFCFG_ARBERR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ARBERR
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-19. Read arbitration Error register EEFCFG_ARBERR bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ARBERR	Arbitration Error Flag '0': No arbitration error occured on Flash memory read '1': An arbitration error occured on Flash memory read The ARBERR bit is set if a read access to the Flash memory cannot be executed because a Flash memory access on the private interface is ongoing The ARBERR bit is cleared by writing '1' to the EEFCFG_ARBCLR:ARBCLR bit If an arbitration error occurs on a read access, then the read data returned will be all '1's.

11.2.19 Read Arbitration Error Register (EEFCFG_ARBCLR)

The Read Arbitration Error Clear Register (EEFCFG_ARBCLR) clears the Arbitration error EEFCFG_ABRERR:ARBERR.

Read arbitration Error Clear register EEFCFG_ARBCLR

Figure 11-20. Read arbitration Error Clear register EEFCFG_ARBCLR

EEFCFG_ARBCLR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ARBCLR
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-20. Arbitration Error Clear register EEFCFG_ARBCLR bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ARBCLR	Arbitration Error Flag Clear Writing '1' clears the arbitration error flag EEFCF-G_ABRERR:ARBERR Writing '0' has no effect

11.2.20 Bus Error Response Register (EEFCFG_BERR)

Bus Error Response Status Register (EEFCFG_BERR) holds the error response cause.

Status Register (EEFCFG_BERR)

Figure 11-21. Status Register (EEFCFG_BERR)

EEFCFG_BERR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WTTM	ACCIGN	ECRWL	UNACC	RESA	RWE	read0	SIZE	CRWE	DED
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R0	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-21. Status Register (EEFCFG_BERR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	WTTM	Arbitration Error Flag Clear Any write transaction on EEFLASH_TABLE_MIR
[8]	ACCIGN	Set to '1' if a Flash memory access is triggered while the previous access is ongoing
[7]	ECRWL	This bit is set if EEFCFG_ECR:ECCOFF is written to more than once or if the Flash configuration unlock sequence is violated while accessing a protected register
[6]	UNACC	Unprivileged write access to the configuration registers except EEFCFG_WCR, EEFCFG_EMENR, EEFCFG_RCR, EEFCFG_SEQWM, EEFCFG_SEQCM, EEFCFG_ARBCLR and EEFCFG_BERRCLR
[5]	RESA	Set to '1' if the reserved address space of the Flash memory and configuration registers is accessed
[4]	RWE	Set to '1' if a protected sector is accessed
[3]	read0	-
[2]	SIZE	Set to '1' if the size of the access to Flash memory is not supported or configuration registers are accessed with an access size of 64 bits.
[1]	CRWE	Set to '1' if a write access to Flash memory is done without setting the EEFCFG_CR:WE bit to '1'
[0]	DED	Double bit ECC fault detected on read access

11.2.21 Bus Error Response Clear Register (EEFCFG_BERRCLR)

The Bus Error Response Clear Register (EEFCFG_BERRCLR).

Status Register (EEFCFG_BERRCLR)

Figure 11-22. Status Register (EEFCFG_BERRCLR)

EEFCFG_BERRCLR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WTTMCLR	ACCIGNCLR	ECRWLCLR	UNACCLR	RESACLR	RWECLR	read0	SIZECLR	CRWECLR	DEDCLR
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0W1	R0W1	R0W1	R0W1	R0W1	R0W1	R0	R0W1	R0W1	R0W1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-22. Status Register (EEFCFG_BERRCLR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	WTTMCLR	Writing '1' clears EEFCFG_BERR:WTTM Writing '0' has no effect
[8]	ACCIGNCLR	Writing '1' clears EEFCFG_BERR:ACCIGN Writing '0' has no effect
[7]	ECRWLCLR	Writing '1' clears EEFCFG_BERR:ECRWL Writing '0' has no effect
[6]	UNACCLR	Writing '1' clears EEFCFG_BERR:UNACC Writing '0' has no effect
[5]	RESACLR	Writing '1' clears EEFCFG_BERR:RESA Writing '0' has no effect
[4]	RWECLR	Writing '1' clears EEFCFG_BERR:RWE Writing '0' has no effect
[3]	read0	-
[2]	SIZECLR	Writing '1' clears EEFCFG_BERR:SIZE Writing '0' has no effect

Table 11-22. Status Register (EEFCFG_BERRCLR) bits

Bit Position	Bit Field Name	Bit Description
[1]	CRWECLR	Writing '1' clears EEFCFG_BERR:CRWE Writing '0' has no effect
[0]	DEDCLR	Writing '1' clears EEFCFG_BERR:DED Writing '0' has no effect

11.3 Operation of the EEPROM Emulation Flash Interface

This chapter describes the operation of EEPROM Emulation Flash Interface.

EEPROM Emulation Flash memory

The EEPROM Emulation Flash macro has up to eight 8KB sectors (refer to the device-specific datasheet for details).

The features supported by the EEPROM Emulation Flash macro include:

- An automatic program algorithm used for write or erase
- A sector erase function (for any combination of sectors)
- The detection of write or erase completion using status bit polling or by CPU interrupts
- The detection of a hang state. For details, refer to 9. Tightly Coupled Flash

EEPROM Emulation Flash memory operation mode

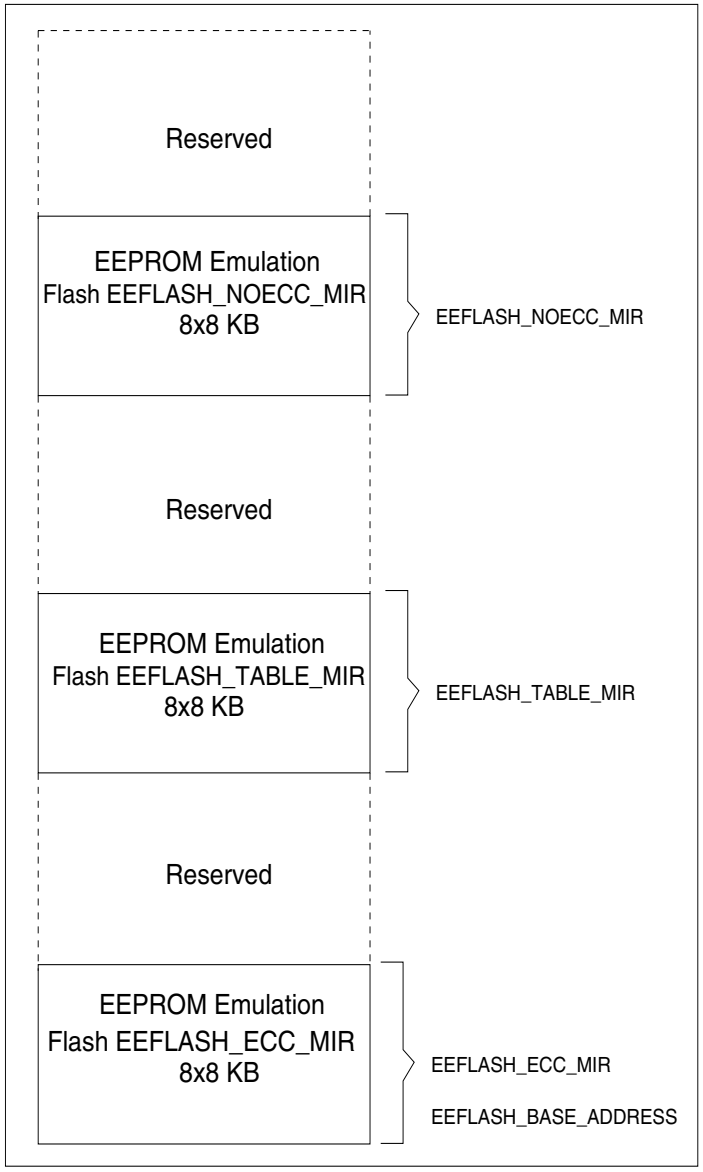
The Flash can be operated in CPU operating mode only. CPU mode can be handled using the command sequencer mode. In this mode, the EEPROM Emulation Flash can be accessed by the CPU or any other master in the system via the EEPROM Emulation Flash Interface. ECC calculation is handled inside the EEPROM Emulation Flash Interface.

Flash write and erase sequences are handled automatically by the command sequencer.

EEPROM Emulation Flash memory address/sector mapping

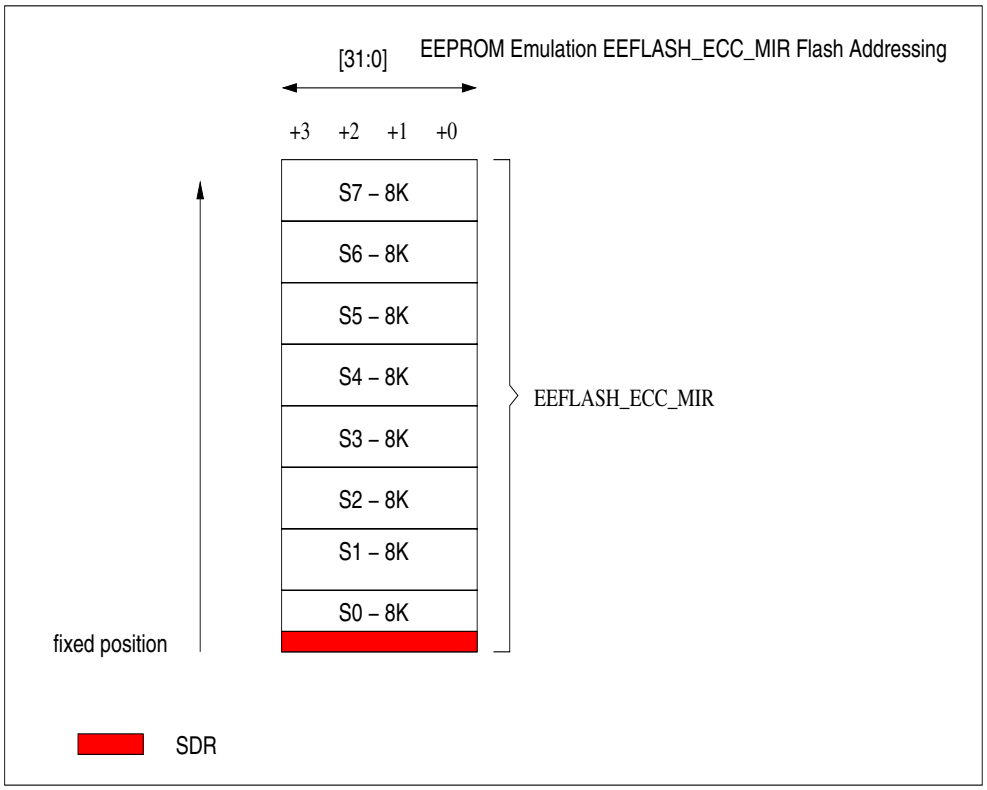
In CPU mode, Flash memory is addressable via three address ranges (refer to the Address mirroring and access control for more details). Address mapping of Flash memory, shown in Figure 11-23 and Figure 11-24, is such that all (8 KB) sectors are kept at a fixed position.

Figure 11-23. EEPROM Emulation Flash address mapping



Note: EEFLASH_BASE_ADDRESS,EEFLASH_ECC_MIR,EEFLASH_TABLE_MIR, and EEFLASH_NOECC_MIR in Figure 11-23 are placed generally. Refer to the datasheet of a particular device to see the actual address.

Figure 11-24. Flash sector/address mapping - CPU mode



Note: Security Description Records (SDR) are located at the lower addresses of S0.

EEPROM Emulation Flash programming in CPU mode

In CPU mode, the Flash write, erase and read/reset commands are handled by the command sequencer. Writing to the Flash must be performed using 32-bit accesses, independent of the ECC configuration.

To program a byte or half-word, the following algorithm must be implemented in software:

- Read the Flash data from the write address
- Modify the read data to include the byte or half-word to be written
- Write the 32-bit data to the write address

Note: Depending on the use case and ECC configuration, an ECC fault may be reported on a read if a word is not completely programmed. In this case, a read shall be issued via the NOECC Mirror.

Enabling ECC calculation in command sequencer mode

Two write modes, 32-bit and 16-bit, are configurable using the EEFCFG_SEQWM:WM16 bit:

- 32-bit write mode
  - In this mode, the command sequencer writes 32-bit data to the target address using two 16-bit write accesses to the Flash memory. The first write to the Flash programs the lower 16 bits and the second write programs the upper 16 bits of write data and ECC parity bits (if enabled)
- 16-bit write mode
  - In this mode, the command sequencer writes 16-bit data to the target address using one 16-bit write access to the Flash memory. If the write access is issued via the NOECC mirror, then the lower 16 bits of write data are programmed without ECC. If the write access is issued via the ECC mirror, then the upper 16 bits and ECC parity bits (if enabled) are programmed to the flash memory

Note: All write accesses in 16-bit write mode must be 32-bit wide because ECC is calculated based on the entire 32-bit write data (32+7 ECC scheme).

Independent of the write mode, ECC is calculated and written by the command sequencer only if:

- Write access is issued via the ECC Mirror AND
- ECC is enabled (i.e. EEFCFG_ECR:ECCOFF= ’0’)

Address mirroring and access control

The EEPROM Emulation Flash is mapped to three different mirroring locations in the memory map. Accessing EEPROM Emulation Flash through this mirroring area is as follows:

- EEFLASH_ECC_MIR: Depending on the EEFCFG_ECR:ECCOFF bit, read and write access by the CPU in this mirror region always checks or generates ECC parity bits
- EEFLASH_TABLE_MIR: This mirror allows only read access by the CPU and will not result in actual read data; software should not access this mirror.
- EEFLASH_NOECC_MIR: This mirror allows read and write accesses without ECC, independent of the EEFCFG_ECR:ECCOFF and ECC calculation enable bits

Note: EEFLASH_ECC_MIR is the default location of the EEPROM Emulation Flash. Mirroring can be disabled once by a hardware reset.

Command sequencer operation

The command sequencer handles all write/program, sector erase and read/reset sequences to the Flash memory. All other commands (i.e. macro erase, sector erase suspend, unlock bypass set, unlock bypass program, unlock bypass reset) are not available.

Specifying addresses

Writing to the EEPROM Emulation Flash in command sequencer mode is possible in 32-bit write accesses only due to the ECC calculation. With 32-bit writes, a 4 byte aligned address must be specified. The EEPROM Emulation Flash Interface calculates ECC for the 32-bit data and programs the 16- or 32-bit data to the Flash, depending on the EEFCFG_SEQCM:WM16 bit. Automatic command sequence and ECC enabling is handled by the interface itself.

Writing and reading data

Writing and reading the EEPROM Emulation Flash is possible only when the command sequencer is in an idle state (EEFCFG_WSR:ST[1:0] = "00"). The command sequencer supports two modes for writing data to Flash memory, both of which are configurable with the EEFCFG_SEQWM:WM16 bit:

- 32-bit write mode (i.e. EEFCFG_SEQWM:WM16 = ‘0’):
  - Write access to EEFLASH_ECC_MIRROR results in the programming of the 32-bit write data, with ECC parity bits calculated from the 32-bit write data to the addressed memory cells
  - Write access to EEFLASH_NOECC_MIRROR results in the programming of the 32-bit write data without ECC parity bits to the addressed memory cells
- 16-bit write mode (EEFCFG_SEQWM:WM16 = ‘1’):
  - Write access to EEFLASH_ECC_MIRROR results in the programming of the upper 16 bits of the 32-bit write data, with ECC parity bits calculated from the 32-bit write data to the addressed memory cells
  - Write access to EEFLASH_NOECC_MIRROR results in programming of the lower 16 bit of the 32-bit write data without ECC parity bits to the addressed memory cells

Independent of the configured write mode, only 32-bit write accesses are accepted by the EEPROM Emulation Interface (EEIF) to the Flash macro.

Data can be read using 8-, 16- and 32-bit access sizes. 64-bit accesses result in a bus error response

Sector erase

Erasing sectors is performed by writing the sector erase command to the two EEFCFG_SEQCM:OPC bits while at least one valid sector for erase is selected using the eight EEFCFG_SEQCM:ERS*E bits.

A valid sector for erase belongs to the interface which requests the erase, i.e. EEIF can not erase SHE Interface (SHEIF) sectors and vice versa, and is not marked as write protected by sector permissions

If multiple sectors are selected in the EEFCFG_SEQCM:ERS*E bits, then erase is performed in descending order.

If the sector erase command is accepted by EEIF, then EEFCFG_SR:RDY is low after writing the EEFCFG_SEQCM:OPC bits. The sector erase is finished when EEFCFG_SR:RDY is high again.

Note: EEIF and SHEIF are explained in greater detail in 11.5 Support for the Secure Hardware Extension (SHE)

Read/reset command

May be used to resolve a Flash macro hang state, which can be caused by:

- Programming a memory cell from ‘0’ to ‘1’
- A time-out of the automatic algorithm because of a Flash macro malfunction

The software can trigger the read/reset command by writing the read/reset command to the EEFCFG_SEQCM:OPC bits.

If the read reset command is accepted by the EEIF, then EEFCFG_SR:RDY is low after writing to the EEFCFG_SEQCM:OPC bits. The read reset is finished when EEFCFG_SR:RDY is high again.

Note: The Flash macro may encounter different hang states during operation which can only be resolved using the Software Flash macro reset.

Software Flash macro reset

To request a reset of the Flash macro, the software can write ‘1’ to the EEFCFG_CR:SWFRST bit. The reset is pending as long as EEIF is not arbitrated. Once EEIF is arbitrated the Flash macro reset is executed immediately. The reset can be used to interrupt an ongoing write or erase operation of EEIF. Once the reset is triggered by writing ‘1’ to the EEFCFG_CR:SWFRST bit another write or erase operation can be requested without waiting for completion of the Flash reset. After writing to the EEFCFG_CR:SWFRST bit, EEFCFG_SR:RDY is low. The Flash macro reset has completed when EEFCFG_SR:RDY is high again.

DMA access using the command sequencer

The command sequencer can only handle one write command at a time. A second write request is ignored and an error response given when a write operation is ongoing. The progress of a write operation can be checked by reading the command sequencer state. Memory blocks can be transferred by using the DMA function. DMA block and burst mode are supported.

Note: It is recommended to use block transfer mode with a block size of 32 bits in the event of a write. A DMA request is generated when the EEFCFG_WCR:DMAEN bit is set to ’1’ and the command sequencer is in an idle state (EEFCFG_WSR:ST[1:0] = ’00’). Any write error during an ongoing transfer stops the DMA and issues an interrupt.

Flash access wait cycles

Based on the system clock frequency and Flash access time (refer to the datasheet for the access time value), the number of wait cycles required for Flash access can be configured by writing to the EEFCFG_CR:FAWC[1:0] bits. By default, these bits are configured for two (2) wait cycles. The same wait cycle settings are applicable for both read and write access to the Flash.

The read data from the Flash is valid only after the configured number of wait cycles. Once the wait states are configured, they continue to apply until they are changed by writing to the Configuration registers. For details on wait cycle settings, refer to Table 11-3.

ECC logic

- The ECC module uses Arm Cortex-R4’s 32-bit ECC scheme for ECC encoding during Flash write and ECC syndrome decoding during Flash read
- During a read operation, the ECC syndrome is computed for the Flash read data syndrome. It is thus interpreted for no error, and correctable and uncorrectable errors. An erased Flash word does not return an ECC error on read
- The ECC module also has an option to inject errors into both the data and the ECC bits read from the Flash to test ECC functionality

- There is an option to disable/switch-off ECC by setting a configuration register bit (i.e. EEFCFCG_ECR:ECCOFF)

Interrupts

The Flash memory interface generates interrupts under the following conditions:

- The Ready Interrupt Flag (i.e. EEFCFG_SR:RDYINT) is set when the Flash memory RDY output goes high (indicating that Flash write, erase or read/reset commands have completed and Flash memory is ready to accept a new command). The interrupt (IRQ) is generated when the EEFCFG_SR:RDYINT interrupt flag and the corresponding Interrupt Enable bit (i.e. EEFCFG_ICR:RDYIE) are set
- The Hang Interrupt Flag (i.e. EEFCFG_SR:HANGINT) is set when the Flash memory HANG output goes high (indicating that the Flash memory has detected a hang condition). The interrupt (IRQ_ERR) is generated when the EEFCFG_SR:HANGINT interrupt flag and the corresponding Interrupt Enable bit (EEFCFG_ICR:HANGIE) are set
- The SEC Interrupt Flag (i.e. EEFCFG_SECIR:SECINT) is set when the ECC checker logic has detected and corrected a single-bit error in the Flash read data. The interrupt (IRQ) is generated if EEFCFG_SECIR:SECINT interrupt flag and the corresponding Interrupt Enable bit (EEFCFG_SECIR:SECIE) is are set

Bus error response

The EEPROM Emulation Flash memory interface generates a bus error response under the following conditions:

- An ECC double or multi-bit error condition
- A write access to Flash memory without the EEFCFG_CR:WE bit set to ’1’
- When write access is anything other than 32 bits, independent of ECC
- Read/write access to a protected sector
- Access to unused Flash memory and Configuration register address space
- Unprivileged write access to the configuration registers except EEFCFG_WCR, EEFCFG_EMENR, EEFCFG_MR, EEFCFG_RCR, EEFCFG_SEQWM, EEFCFG_SEQCM, EEFCFG_ARBCLR, EEFCFG_BERRCLR
- Writing to the EEFCFG_ECR register more than once
- Any deviation in the Flash configuration unlock sequence
- In command sequencer mode when the current access is ongoing and the next one is ignored
- Any write transaction on EEFLASH_TABLE_MIR.

Flash security

To prevent unauthorized reading of the Flash macros and access protection during application, run time Flash security is introduced. Flash security is distributed between the Flash Interface and Security Checker modules.

The Flash Interface receives EEPROM Emulation Flash sector-wise read and write access permissions from the Security Checker module and performs the following tasks:

- Access is allowed to the addressed sector if permission is granted
- If read/execute access is denied to the addressed sector
  - The Flash Interface prevents access to the Flash and default data is output onto the bus
  - A bus error response is generated to inform the master
- If write access is denied to the addressed sector
  - The Flash memory falls back to read mode if the program/command sequence points to a protected sector. If, however, the program sequence is correct but the actual write address (i.e. PA/SA) points to a protected sector, the Flash macro might enter a HANG state
  - A bus error response is generated to inform the master
- Flash access is completely blocked during a scan test to avoid scanning out the Flash contents. This is done by forcing the Flash internal supply reset to ‘0’ during a scan test mode, causing the Flash macro to enter sleep state

Note: All read, write or execute accesses to Flash memory are checked against the received permission settings.

Sector and macro erase protection

- Sector erase is disabled for a write protected sector

- In CPU mode, the Flash macro erase is disabled if at least one Flash sector is write protected.
- In FPP mode, the Flash macro erase is disabled if the chip erase disable marker (inside the SDR) is written with a specific value (all other values will allow the macro erase operation). Also if chip erase is disabled, any write operation to the Flash will also be discarded

Special Permissions for BDR & SDR

- Sectors storing a Boot Description Record (BDR) and SDR will have special and different access permissions and hence they are stored in different sectors

Note:

- In command sequencer mode, a read permission to the respective sector should be enabled, otherwise an error interrupt will be generated
- Configuration details of Flash sector-wise access permissions can be found in the SDR layout of the BOOTROM Software Interface (16. Boot ROM Software Interface)
- For active permission set selection details refer to the Security Checker module in 5. Security Checker
- When the Flash macro enters a HANG state, it can be brought back to normal command state by: a read/reset command; a software triggered Flash reset (i.e. TCFCFG_FCFGR:SWFRST); a software soft reset; or a hardware reset

11.4 Notes on using Flash memory

This section contains notes on using Flash Memory. For more details on Flash Memory, refer to 9. Tightly Coupled Flash and the Confirming the automatic algorithm execution state on page 526 and Writing to and erasing Flash memory on page 533.

Input of a CPU reset

A CPU reset (power reset, external reset, software reset, watchdog reset or clock stop reset, software triggered hardware reset, clock supervisor reset, PD3 reset) asserts the Flash hardware reset. A reset assertion during Flash writing causes the data written to be undefined. Resetting the device once execution of a sector erase has begun corrupts the data in the sector. In that case, restart the erase on this sector and allow it to complete. If a reset occurred during writing, rewrite with the same data.

Register configuration

Since a Flash reset is asserted for a minimum period of 500 ns during a software triggered Flash reset, the software should wait until the Flash reset has completed before starting memory access. To know that the reset has completed and the Flash is ready for the next command, the software can read the EEFCFG_SR:RDY bit (the RDY output from Flash macro returns to ’1’ after a time period that does not exceed 80 μs).

In addition, the CAM output from the Flash macro is not valid during reset assertion and for 70 ns after reset release. Thus it should be ensured that the TCFCFG_FCAMLRn and TCFCFG_FCAMHRn registers (in the Tightly Coupled Flash) are not accessed during this period.

Program access to Flash memory

While the Flash memory is being erased or data written to it, reading is not possible. Hence, when it is required to read data from the Flash memory while the memory is being erased or written to, the required data must first be copied to RAM before starting the erase/write operation.

Flash hardware reset

Configuring EEFCFG_CR:SWFRST to ’1’ asserts the Flash macro reset. The software should check the status of the Flash by reading EEFCFG_SR:RDY. Until EEFCFG_SR:RDY equals ‘1’, the software should not issue the next access to Flash memory.

11.5 Support for the Secure Hardware Extension (SHE)

This section contains notes on using the EEPROM Emulation Flash Memory Interface for devices that are equipped with a Secure Hardware Extension (SHE).

General

The SHE IP requires a non-volatile memory to store cryptographic keys. For this reason the EEPROM Emulation Flash is shared between the EEPROM emulation and the SHE IP. Sectors six and seven are exclusively reserved for SHE, and this area is subtracted from the EEPROM emulation space.

The EEPROM Emulation Flash supporting a SHE IP contains two bus interfaces:

- The EEPROM Emulation Interface (EEIF)
- The SHE Interface (SHEIF)

Hardware barrier

A hardware barrier, as shown in Figure 11-25, is implemented to avoid accesses from the EEIF to Flash sectors reserved for the SHEIF and vice versa. Any attempt from the EEIF to access a Flash sector protected by the hardware barrier results in a bus error response on the bus. In addition the EEFCFG_BERR:RESA register bit is set.

Separate register sets

A number of additional registers are provided by the EEFLASH Emulation Flash with SHE support. The EEIF and SHEIF implement independent sets of configuration and status registers.

The settings configured in the EEIF configuration registers only apply to accesses issued by the EEIF.

Status information in the EEIF status registers is only updated if they are related to an access by EEIF.

Arbitration of Flash macro accesses

An arbiter is implemented so that both the EEIF and SHEIF can access to the shared Flash macro. The arbiter uses a fixed priority scheme and evaluates arbitration after each single transfer. SHEIF has a higher priority than EEIF.

For example, if the Flash macro is read by an AHB burst with length 16, arbitration is evaluated after each burst transfer. This implies that the read burst may be interrupted by an access to the Flash macro on SHEIF.

All Flash macro accesses involving the command sequencer (write, sector erase, read reset) behave the same independent of the arbitration state except for the run time required for the command execution. For example, if a valid write access is driven on the EEIF but SHEIF is currently arbitrated, then the access is accepted by the EEPROM Emulation Flash but its execution is stalled until the SHEIF access has finished. Once the command sequencer starts the execution of the command, the arbitration is kept at the EEIF until the command execution is finished.

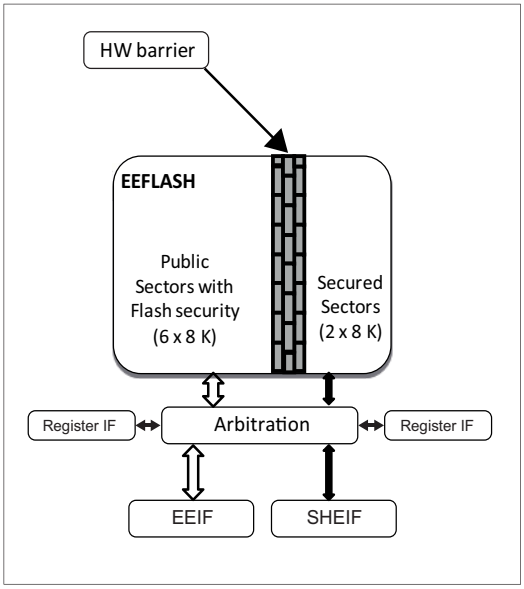
Read accesses by the EEIF behave differently depending on the arbitration configuration and state:

- If EEFCFG_EMENR:AEE=0, a read access stalls the bus until it is arbitrated and then returns Flash macro read data
- If EEFCFG_EMENR:AEE=1:
  - If EEIF is arbitrated, then Flash macro read data is returned
  - If SHEIF is arbitrated, the EEFCFG_ARBERR:ARBERR flag is set to ‘1’ to indicate an arbitration error and read data returns all ‘1’s

This non-blocking read access is required to avoid stalling the bus in case a time consuming SHEIF operation is running, such as a write or sector erase.

The application software must check the EEFCFG_ARBERR:ARBERR flag after one or multiple read accesses to determine whether valid data was returned. This further implies that code execution from the EEPROM Emulation Flash is not possible.

Figure 11-25. HW barrier to isolate EEIF (public) and SHEIF (SHE) data paths



12. TCM RAM Interface

This chapter explains the function and operation of the TCM RAM Interface (TCMRAM_IF).

12.1 Outline of the TCMRAM_IF

The Tightly Coupled Memory (TCM) RAM Interface (IF) module (hereafter called TCMRAM_IF) acts as an interface between the Arm® Cortex® -R4 processor and the TCM.

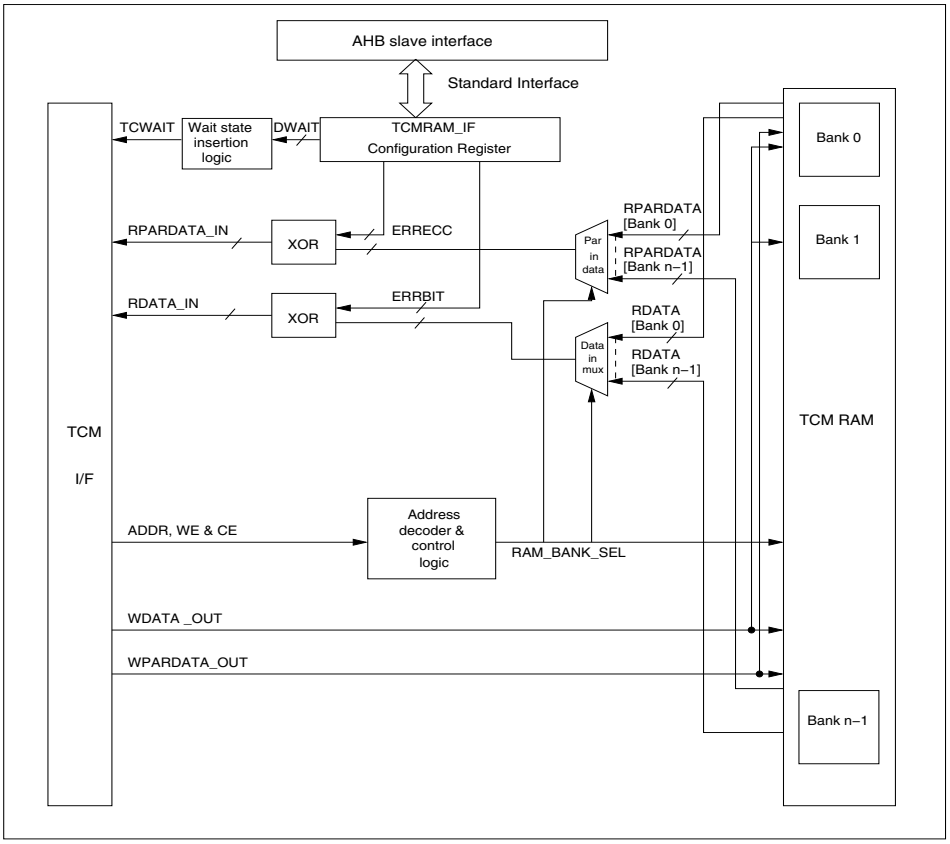
Features of the TCMRAM_IF

The features of the TCMRAM_IF module are:

- It connects a RAM to the Tightly Coupled Memory (TCM) port of the Arm Cortex-R4 processor. For details of TCM port, refer to the Sections 11.4 (About the TCMs) and 9.6.1 (TCM RAM access) of Arm_Cortex-R4 _TechnicalReferenceManual.pdf (refer to http://infocenter.arm.com)
- Error injection using data read from the TCM RAM to test the ECC functionality enabled in Arm Cortex-R4 processor. Either the data, or the ECC data, or both the data and the ECC data read from the TCM RAM can be corrupted to inject the errors
- Error generation due to non-privileged and unsupported accesses of the control registers

Block diagram of TCMRAM_IF

Figure 12-1. Block diagram of TCMRAM_IF



TCWAIT : Wait signal input to TCM interface of the processor

RPARDATA_IN : Input ECC data from the TCM RAM blocks to the TCM Interface of the processor

RDATA_IN : Input data from the TCM RAM blocks to the TCM Interface of the processor

WDATA_OUT : Output data from the processor to the TCM RAM banks

WPARDATA_OUT : Output ECC data from the processor to the TCM RAM banks

RPARDATA [bankn] : Parity data from the TCM RAM bankn

RDATA[bankn] : Data from the TCM RAM bankn

RAM_BANK_SEL : Output of the 'address decoder and control logic' used to select the correct data from the RAM banks

ADDR, WE, and CE : Outputs from the TCM Interface of the processor (address, write enable, and chip enable)

12.2 TCMRAM_IF registers

The TCMRAM_IF has three registers which can be read in both privileged and non-privileged mode. Writing, however, can only be done in privileged mode through the configuration bus interface.

Registers of TCMRAM_IF

The following registers are available for the TCMRAM_IF:

- TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0)
- TCMRAM_IF Configuration Register 1 (TRCFG_TCMCFG1)
- TCMRAM_IF UNLOCK Register (TRCFG_TCMUNLOCK)

Memory layout of TCMRAM_IF registers

Table 12-1. Memory layout of TCMRAM_IF registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	TRCFG_TCMCFG1 00000000 00000000 00000000 00000000				TRCFG_TCMCFG0 00000011 00000000 00000001 00000000			
0x00000008	read0 00000000 00000000 00000000 00000000				TRCFG_TCMUNLOCK 00000000 00000000 00000000 00000000			

12.2.1 TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0)

The TCMRAM_IF Configuration Register (TRCFG_TCMCFG0) holds 7-bit ERRECC data, a 2-bit wait state value and a LOCKSTATUS bit indicating whether the configuration registers are locked or unlocked.

TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0)

Figure 12-2. TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0)

TRCFG_TCMCFG0																													
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**
read0	read0	read0	read0	read0	read0	DWAIT[1]	DWAIT[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	LOCKSTATUS	read0	ERRECC[6]	ERRECC[5]	ERRECC[4]	ERRECC[3]	ERRECC[2]	ERRECC[1]	ERRECC[0]
R0	R0	R0	R0	R0	R0	RWP	RWP	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 12-2. TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	DWAIT	Data WAIT State Value These two bits are used to configure the number of data wait states based on the CLK_SYS_PD3 clock. '00': Zero wait state '01': One wait state '10': Two wait states '11': Three wait states **Note:** The wait state value is applicable for read accesses only.
[23:16]	read0	-

Table 12-2. TCMRAM_IF Configuration Register 0 (TRCFG_TCMCFG0) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	LOCKSTATUS	TCMRAM_IF Lock Status This bit provides the locked/unlocked status of TCMRAM_IF. '0': TCMRAM_IF Configuration Registers are unlocked '1': TCMRAM_IF Configuration Registers are locked By default this bit is high; writing the correct unlock value to the TRCFGn_TCMUNLOCK register resets this bit to '0' and writing the correct lock value sets this bit to '1'.
[7]	read0	-
[6:0]	ERRECC	ECC ERRBIT Value These bits are used to corrupt the Error Correction Code (ECC) bits read from the RAM.

12.2.2 TCMRAM_IF Configuration Register 1 (TRCFG_TCMCFG1)

The TCMRAM_IF Configuration Register 1 holds the 32-bit ERRBIT data used to corrupt the data read from the TCM RAM. This feature can be used to test the ECC logic.

TCMRAM_IF Configuration Register 1 (TRCFG_TCMCFG1)

Figure 12-3. TCMRAM_IF Configuration Register 1 (TRCFG_TCMCFG1)

TRCFG_TCMCFG1																															
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**
ERRBIT[31]	ERRBIT[30]	ERRBIT[29]	ERRBIT[28]	ERRBIT[27]	ERRBIT[26]	ERRBIT[25]	ERRBIT[24]	ERRBIT[23]	ERRBIT[22]	ERRBIT[21]	ERRBIT[20]	ERRBIT[19]	ERRBIT[18]	ERRBIT[17]	ERRBIT[16]	ERRBIT[15]	ERRBIT[14]	ERRBIT[13]	ERRBIT[12]	ERRBIT[11]	ERRBIT[10]	ERRBIT[9]	ERRBIT[8]	ERRBIT[7]	ERRBIT[6]	ERRBIT[5]	ERRBIT[4]	ERRBIT[3]	ERRBIT[2]	ERRBIT[1]	ERRBIT[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-3. TCMRAM_IF Configuration Register 1 (TRCFG_TCMCFG1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ERRBIT	ERRBIT Value This register holds the 32-bit ERRBIT data used to corrupt both the lower and upper words of the TCM data read from the TCM RAM.

12.2.3 TCMRAM_IF UNLOCK Register (TRCFG_TCMUNLOCK)

The TCMRAM_IF UNLOCK Register has to be written with the correct unlock value (0xACC55ECC) to reset the TRCFG_TCMCFG0:LOCKSTATUS bit value so that the two configuration registers (TRCFG_TCMCFG0 and TRCFG_TCMCFG1) can be written to. After configuration, this register has to be written with the correct lock value (0x5ECCB10C) which sets the TRCFG_TCMCFG0:LOCKSTATUS bit and restricts the write access to the two configuration registers. This register can only be written to with 32-bit or 64-bit privileged accesses through the configuration bus interface.

TCMRAM_IF Unlock Register (TRCFG_TCMUNLOCK)

Figure 12-4. TCMRAM_IF Unlock Register (TRCFG_TCMUNLOCK)

TRCFG_TCMUNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-4. TCMRAM_IF Unlock Register (TRCFG_TCMUNLOCK)

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	TCMRAM_IF Unlock The TCMRAM_IF Unlock Register protects the TCMRAM_IF interface registers from being accidently modified by software. The static configuration registers cannot be written to until this register contains a specific value, which can only can be written in privilege mode. To lock the TCMRAM_IF Configuration Registers again, the software must write another specific value. Illegal access to static configuration registers or writing incorrect lock or unlock values to this register causes a protection error. Reading this register always returns '0'.

12.3 Operation of the TCMRAM_IF

This chapter describes the operation of the TCMRAM_IF.

Wait states

Wait states can be configured for this module by writing to the TRCFG_TCMCFG0 register through the configuration bus interface in privileged mode and only when the TRCFG_TCMCFG0:LOCKSTATUS bit is zero. Zero to a default maximum of three wait states can be configured. The wait states are inserted based on the Arm Cortex-R4 clock.

The data read from the RAM is sampled by the TCM port of the processor only after the number of wait states is configured. This number is valid until the value in the configuration registers is changed.

ECC test

This interface has a facility to inject errors into the data read from the RAM to test the 32-bit ECC feature enabled in the Arm Cortex-R4 processor.

Introducing errors through data corruption is achieved by writing data (which corrupts the data read from RAM) to TRCFG_TCMCFG0 and TRCFG_TCMCFG1 (the configuration registers) when the TRCFG_TCMCFG0:LOCKSTATUS bit is zero. To be more specific, 32-bit ERRBIT data has to be written to the TRCFG_TCMCFG1 Register. Or if the errors are to be introduced by corrupting the ECC data, then 7-bit TRCFG_TCMCFG0:ERRECC data has to be written to the TRCFG_TCMCFG0 register.

Note: During a write operation to TCM RAM, the data is written directly from the TCM port of the Arm Cortex-R4 to TCM RAM. However, during a read operation from the TCM RAM, both the lower and upper 32 bits of the 64-bit data read are XOR-ed with the 32-bit ERRBIT data in TRCFG_TCMCFG1. The ECC bits corresponding to both words are XOR-ed with the 7-bit TRCFG_TCMCFG0:ERRECC data. During normal operation, the software has to ensure that these bits are zero.

Error generation

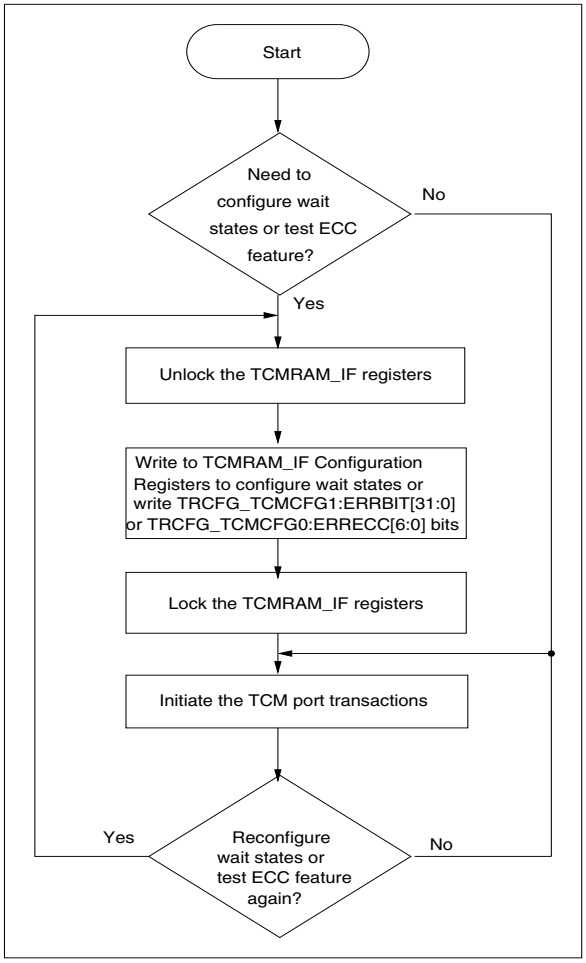
The TCMRAM_IF generates an error when any of the following condition occurs:

- A write access to the TCMRAM_IF Configuration Registers before they are unlocked
- When a value other than the unlock or lock value is written to the TRCFG_TCMUNLOCK register
- A non-privileged write access to the TCMRAM_IF registers
- A read/write access to the reserved space of the module
- A write occurs to a read-only register (attribute: read0)
- A write to TRCFG_TCMUNLOCK register with anything other than 32 or 64-bit accesses

12.4 TCMRAM_IF flowchart

The following flowchart shows the sequence of steps required to configure the wait states or enable the ECC test feature.

Figure 12-5. Configuration flowchart



13. SRAM Interface

This chapter explains the function and operation of the Static RAM Interface (SRAM_IF).

13.1 Outline of the SRAM_IF

This section describes the features and the block diagram of the Static RAM Interface, hereafter called the SRAM_IF.

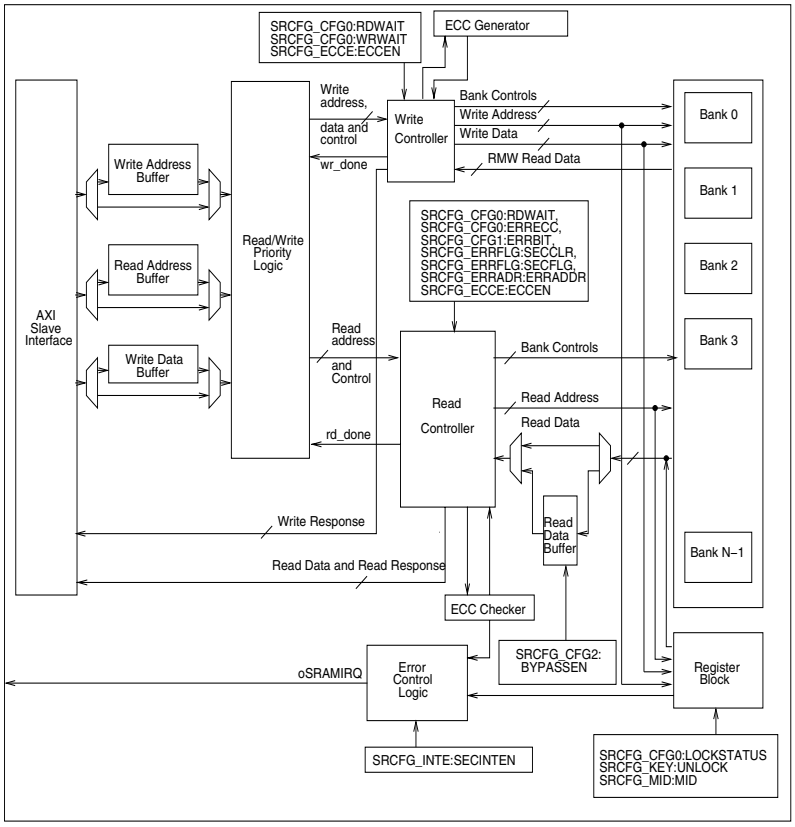
Features of the SRAM_IF

The SRAM_IF module interacts with the system SRAM macro. It has Error Correction Code (ECC) logic, which can generate interrupts during bit-error detection in SRAM read data. The ECC logic can also act as a form of tester in that it has the option of injecting bit errors into SRAM read data. The SRAM_IF:

- Integrates synchronous static RAM
- Implements single-bit error correction and double-error detection ECC logic similar to the Arm Cortex-R4 ECC implementation
- Generates an interrupt during single-bit error detection
- Sends an error response in the event of double-bit error detection
- Injects errors to test ECC functionality by changing both the data and ECC bits read from the SRAM
- Inserts wait states during an SRAM read and write operation so that the memory access time can be adjusted to higher operating frequencies
- Supports parallel read/write either to two different SRAM banks or to a single SRAM bank and register set
- Implements a Least Recently Granted (LRG) priority scheme in case of simultaneous read and write requests to the same SRAM bank or register set

Block diagram of SRAM_IF

Figure 13-1. Block diagram of SRAM_IF



13.2 SRAM_IF registers

This section describes the SRAM_IF registers.

Registers of SRAM_IF

The following registers are available for the SRAM_IF:

- SRAM_IF Configuration Register 0 (SRCFG_CFG0)
- SRAM_IF Configuration Register 1 (SRCFG_CFG1)
- SRAM_IF Configuration Register 2 (SRCFG_CFG2)
- SRAM_IF Unlock/Lock Key Register (SRCFG_KEY)
- SRAM_IF Error Flag Register (SRCFG_ERRFLG)
- SRAM_IF Interrupt Enable Register (SRCFG_INTE)
- SRAM_IF ECC Enable Register (SRCFG_ECCE)
- SRAM_IF Error Address Register (SRCFG_ERRADR)
- SRAM_IF Module Identification Register (SRCFG_MID)

Memory layout of SRAM_IF registers

Table 13-1. Memory layout of SRAM_IF registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	SRCFG_CFG1 00000000 00000000 00000000 00000000				SRCFG_CFG0 00000011 00000011 00000001 00000000			
0x00000008	SRCFG_KEY 00000000 00000000 00000000 00000000				SRCFG_CFG2 00000000 00000000 00000000 00000000			
0x00000010	SRCFG_INTE 00000000 00000000 00000000 00000000				SRCFG_ERRFLG 00000000 00000000 00000000 00000000			
0x00000018	read0 00000000 00000000 00000000 00000000				SRCFG_ECCE 00000000 00000000 00000000 00000001			
0x00000020	SRCFG_MID 00000000 00000000 00000000 00000000				SRCFG_ERRADR 00000000 00000000 00000000 00000000			

13.2.1 SRAM_IF Configuration Register 0 (SRCFG_CFG0)

The SRAM_IF Configuration Register 0 contains the 7-bit SRCFG_CFG0:ERRECC data used to inject errors into the ECC data read from SRAM. It also contains 2-bit wait state values as well as the lock/unlocked status bit, SRCFG_CFG0:LOCK-STATUS.

SRAM_IF Configuration Register 0 (SRCFG_CFG0)

Figure 13-2. SRAM_IF Configuration Register 0 (SRCFG_CFG0)

SRCFG_CFG0																													
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**
read0	read0	read0	read0	read0	read0	RDWAIT[1]	RDWAIT[0]	read0	read0	read0	read0	read0	read0	WRWAIT[1]	WRWAIT[0]	read0	read0	read0	read0	read0	read0	read0	LOCKSTATUS	read0	ERRECC[6]	ERRECC[5]	ERRECC[4]	ERRECC[3]	ERRECC[2]	ERRECC[1]	ERRECC[0]
R0	R0	R0	R0	R0	R0	RWP	RWP	R0	R0	R0	R0	R0	R0	RWP	RWP	R0	R0	R0	R0	R0	R0	R0	R	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 13-2. SRAM_IF Configuration Register 0 (SRCFG_CFG0) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	RDWAIT	Read Data Wait State Value These bits are used to configure the number of data wait states for an SRAM read transaction. '00': 0 wait states '01': 1 wait state '10': 2 wait states '11': 3 wait states (default)
[23:18]	read0	-
[17:16]	WRWAIT	Write Data Wait State Value These bits are used to configure the number of data wait states for an SRAM write transaction. '00': 0 wait states '01': 1 wait state '10': 2 wait states '11': 3 wait states (default)

Table 13-2. SRAM_IF Configuration Register 0 (SRCFG_CFG0) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	LOCKSTATUS	SRAM_IF Lock Status This bit provides the status, i.e. locked or unlocked, of the SRAM_IF. '0': SRAM_IF is unlocked and the configuration registers are accessible for write or read operations '1': SRAM_IF is locked and the configuration registers are not accessible for a write operation(default)
[7]	read0	-
[6:0]	ERRECC	ECC ERRBIT Value Each bit set in this register causes an emulated bit-flip in the corresponding bit of read ECC bits.

Notes:

1. The seven SRCFG_CFG0:ERRECC bits (bits [6:0]) - have no effect when ECC is disabled (i.e. SRCFG_ECCE:ECCEN = '0').
2. In the event of a 64-bit read transfer, the seven SRCFG_CFG0:ERRECC bits cause emulated bit-flips in the ECC bits of both the upper and lower 32-bit read data.

13.2.2 SRAM_IF Configuration Register 1 (SRCFG_CFG1)

The SRAM_IF Configuration Register 1 contains the 32-bit SRCFG_CFG1:ERRBIT data used to corrupt the data read from the SRAM.

SRAM_IF Configuration Register 1 (SRCFG_CFG1)

Figure 13-3. SRAM_IF Configuration Register 1 (SRCFG_CFG1)

SRCFG_CFG1																															
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**
ERRBIT[31]	ERRBIT[30]	ERRBIT[29]	ERRBIT[28]	ERRBIT[27]	ERRBIT[26]	ERRBIT[25]	ERRBIT[24]	ERRBIT[23]	ERRBIT[22]	ERRBIT[21]	ERRBIT[20]	ERRBIT[19]	ERRBIT[18]	ERRBIT[17]	ERRBIT[16]	ERRBIT[15]	ERRBIT[14]	ERRBIT[13]	ERRBIT[12]	ERRBIT[11]	ERRBIT[10]	ERRBIT[9]	ERRBIT[8]	ERRBIT[7]	ERRBIT[6]	ERRBIT[5]	ERRBIT[4]	ERRBIT[3]	ERRBIT[2]	ERRBIT[1]	ERRBIT[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-3. SRAM_IF Configuration Register 1 (SRCFG_CFG1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ERRBIT	ERRBIT Value Each bit set in this register causes an emulated bit-flip in the corresponding bit of the 32-bit read data.

Notes:

1. All subsequent accesses will fail if ECC injection is configured.
2. All 32 SRCFG_CFG1:ERRBIT bits have no effect if the ECC is disabled (i.e. SRCFG_ECCE:ECCEN = '0').
3. In the event of a 64-bit read transfer, the SRCFG_CFG1:ERRBIT bits cause emulated bit-flips in both the upper and lower 32-bit read data.

13.2.3 SRAM_IF Configuration Register 2 (SRCFG_CFG2)

The SRAM_IF Configuration Register 2 contains the SRCFG_CFG2:BYP ASSEN bit which is used to enable the Read Data Buffer (RDB) bypass path for data read from the SRAM.

SRAM_IF Configuration Register 2 (SRCFG_CFG2)

Figure 13-4. SRAM_IF Configuration Register 2 (SRCFG_CFG2)

SRCFG_CFG2																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	BYPASSEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-4. SRAM_IF Configuration Register 2 (SRCFG_CFG2) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	BYPASSEN	RDB Bypass Disable/Enable '0': RDB bypass path is disabled (default) '1': RDB bypass path is enabled

Note: Enabling the RDB bypass path (i.e. SRCFG_CFG2:BYPASSEN = '1') reduces read latency since the data read from SRAM will be directly output to the bus interface without being buffered to the RDB (refer to Figure 13-1). For the FCR4 Cluster series, SRCFG_CFG2:BYPASSEN can be set to '1' only if the 2-bit SRCFG_CFG0:RDWAIT > '00'.

13.2.4 SRAM_IF Unlock/Lock Key Register (SRCFG_KEY)

The SRAM_IF Unlock/Lock Key Register has to be written with the value 0x5ECC551F to unlock or 0x551FB10C to lock all accesses to the SRAM_IF Configuration Registers.

SRAM_IF Unlock/Lock Key Register (SRCFG_KEY)

Figure 13-5. SRAM_IF Unlock/Lock Key Register (SRCFG_KEY)

SRCFG_KEY																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-5. SRAM_IF Unlock/Lock Key Register (SRCFG_KEY) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	SRAM_IF Unlock/Lock Key If the correct unlock key is written to this register, the SRCFG_CFG0:LOCKSTATUS is set to '0' (unlocked) and all the configuration registers can be written to. If the correct lock key is written to this register, the SRCFG_CFG0:LOCKSTATUS is set to '1' (locked) and the configuration registers will not be accessible. Illegal access to static configuration registers or writing a value other than the lock or unlock value to this register causes a protection error and an error response is returned.

13.2.5 SRAM_IF Error Flag Register (SRCFG_ERRFLG)

The SRAM_IF Error Flag Register contains the detection flag that indicates a single error in SRAM read data. It also contains the bit to clear this flag.

SRAM_IF Error Flag Register (SRCFG_ERRFLG)

Figure 13-6. SRAM_IF Error Flag Register (SRCFG_ERRFLG)

SRCFG_ERRFLG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SECCLR	read0	read0	read0	read0	read0	read0	read0	SECFLG
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-6. SRAM_IF Error Flag Register (SRCFG_ERRFLG) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8]	SECCLR	Single-Bit Error Flag Clear If this bit is '1', it clears the SRCFG_ERRFLG:SECFLG flag. Reading this bit always returns '0'.
[7:1]	read0	-
[0]	SECFLG	Single-Bit Error Detection Flag This bit is set if a single-bit error is detected in SRAM read data. This also initiates the sending of a maskable interrupt to the CPU. If the SRCFG_INTE:SECINTEN bit is '0', no interrupt occurs.

Note: An ECC check is performed in parallel on both the upper and lower 32-bits of SRAM read data. Irrespective of the access size, SRCFG_ERRFLG:SECFLG is set when a single-bit error is detected in either the upper or lower 32-bits of SRAM read data.

13.2.6 SRAM_IF Interrupt Enable Register (SRCFG_INTE)

The SRAM_IF Interrupt Enable Register contains the interrupt enable bits.

SRAM_IF Interrupt Enable Register (SRCFG_INTE)

Figure 13-7. SRAM_IF Interrupt Enable Register (SRCFG_INTE)

SRCFG_INTE																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SECINTEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-7. SRAM_IF Interrupt Enable Register (SRCFG_INTE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	SECINTEN	Single-Bit Error Interrupt Enable bit '0': An interrupt will not be generated even if the SRCFG_ERRFLG:SECFLG bit is set (default) '1': An interrupt will be generated if the SRCFG_ERRFLG:SECFLG bit is set (single error detection)

13.2.7 SRAM_IF ECC Enable Register (SRCFG_ECCE)

The SRAM_IF ECC Enable Register contains the ECC enable bits.

SRAM_IF ECC Enable Register (SRCFG_ECCE)

Figure 13-8. SRAM_IF ECC Enable Register (SRCFG_ECCE)

SRCFG_ECCE																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ECCEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 13-8. SRAM_IF ECC Enable Register (SRCFG_ECCE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ECCEN	ECCEN Value This bit enables/disables the ECC logic in the SRAM_IF. It is writable only once by software; further writes to this bit will be signaled as a protection error and an error response will be returned. '0': ECC logic is disabled '1': ECC logic is enabled (default)

13.2.8 SRAM_IF Error Address Register (SRCFG_ERRADR)

The SRAM_IF Error Address Register contains the absolute SRAM location address of where a single-bit error has occurred. An ECC check is always performed in parallel on both the upper and lower 32-bits of SRAM read data, and independent of the error location, the 64-bit aligned address is stored in this register.

SRAM_IF Error Address Register (SRCFG_ERRADR)

Figure 13-9. SRAM_IF Error Address Register (SRCFG_ERRADR)

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

Table 13-9. SRAM_IF Error Address Register (SRCFG_ERRADR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ERRADDR	ECC Error Address This read-only register contains the absolute SRAM address of where a single-bit ECC error has occurred.

13.2.9 SRAM_IF Module Identification Register (SRCFG_MID)

The SRAM_IF Module Identification Register contains the SRAM_IF module ID.

SRAM_IF Module Identification Register (SRCFG_MID)

Figure 13-10. SRAM_IF Module Identification Register (SRCFG_MID)

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

Table 13-10. SRAM_IF Module Identification Register (SRCFG_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID This read-only register gives the unique identification (ID) number of the SRAM_IF module. The ID number identifies the version of the SRAM_IF module used in the MCU and can be found in the device-specific datasheet.

13.3 Operation of the SRAM_IF

This section describes the features supported by the SRAM_IF.

Wait states

Wait states can be configured by writing to the SRCFG_CFG0 register through the AXI slave interface in privileged mode and only when the SRCFG_CFG0:LOCKSTATUS bit is '0'. An error response will be generated if there is a write to SRCFG_CFG0:RDWAIT or SRCFG_CFG0:WRWAIT when SRCFG_CFG0:LOCKSTATUS is '1'. The range of wait states that can be configured range from 0 to 3. By default, three (3) wait states are configured, which means the data will be latched on the fourth clock cycle after the address is put onto the memory interface for a read transaction. Similarly, for a write transaction, data will latched after a configured number of clock cycles specified by the SRCFG_CFG0:WRWAIT bits.

The data read from the SRAM is used only after the configured number of wait states are completed. Once the wait states are configured, they continue to apply until they are changed by writing to the configuration registers.

The SRCFG_CFG0:RDWAIT is used by the write controller in case of a Read-Modified Write (RMW; See “ECC operation” below), and by the read controller for read transactions. These blocks will generate the wait states as per configuration and latch the memory data accordingly.

ECC operation

The ECC generator calculates the ECC for write transactions and ECC checker checks the ECC for read transactions.

When enabled, ECC checks will be carrired out on all data read from the SRAM. This means that single-bit errors would be detected and corrected, while double-bit errors would be detected and an error response sent to the master. Similarly, if ECC is enabled, it will be calculated for every write transaction. Refer to the Arm Cortex-R4 for details on ECC generation and ECC checking algorithms.

Partial writes occur when write transactions write only part of the memory address (8- or 16-bit). The ECC for these writes will be calculated after the new data has been merged with old valid data in memory (other than the 8- or 16-bit write data). This operation is called Read-Modified Write (RMW). In other words, the memory data will be read for the partial write address, and then written after merging the partial write data and recalculating the ECC. In order to avoid the RMW penalty, it is recommended to store 8- and 16-bit data in SRAM as 32 bits.

Disabling the ECC improves performance for partial writes because the clock cycles used to read the data from memory and recalculate the ECC will be saved. Instead, the write will be executed in one cycle.

ECC self-test

The ECC self-test feature is implemented to test the 32-bit ECC logic. This is done by injecting bit-flips into either the data read from the SRAM, the ECC data or both.

This is achieved by writing the required data into the SRAM_IF Configuration Register when the SRCFG_CFG0:LOCKSTATUS bit is '0' (Accessing this register when SRCFG_CFG0:LOCKSTATUS is '1' will generate an error response). If the errors are to be injected into the data, then 32-bit data has to be written to SRCFG_CFG1:ERRBIT, and if the errors are to be injected into the ECC data, then 7-bit data has to be written to SRCFG_CFG0:ERRECC. In the event of a 64-bit read transfer, the 32 SRCFG_CFG1:ERRBIT bits cause emulated bit-flips in both the upper and lower 32-bit data, and the seven SRCFG_CFG0:ERRECC bits cause emulated bit-flips in the ECC bits of both the upper and lower 32-bit data.

Note: The data (32-bit data and 7-bit ECC data) read from RAM locations are always flipped when SRCFG_CFG0:ERRECC and SRCFG_CFG1:ERRBIT are '1'. If this feature is not required, the register values will be '0', i.e. the default value.

Error generation

An error response is generated when:

- The SRAM_IF Configuration Registers are accessed before they are unlocked
- A value other than the correct unlock or lock keys are written to the SRCFG_KEY register
- The SRAM_IF Configuration Registers are accessed in non-privileged mode
- Access is attempted to the reserved space of the module

The usage of SRCFG_KEY:UNLOCK to access the register space is given in Figure 13-11. The memory map and reserved spaces are shown in Table 13-1.

Interrupt generation

SRAM_IF generates interrupts under the following condition:

- The ECC checker has detected a single-bit error in the SRAM read data (i.e. SRCFG_ERRFLG:SECFLAG has been set) and the SRCFG_INTE:SECINTEN bit is set. SRAM_IF also stores the address of the SRAM location of where the single-bit error has occurred in the SRCFG_ERRADR:ERRADDR register

Parallel read/write access and priority scheme

The SRAM_IF AXI interface is shared for both memory and register accesses. The SRAM_IF supports parallel read and write accesses to either two different SRAM banks or a single SRAM bank and register set. In the event of simultaneous read and write requests to the same SRAM bank or register set, the Least Recently Granted (LRG) priority scheme is used to grant (or permit) one of these transactions. If the last granted request was a read, then a write request will be given priority and vice-versa. After a reset, a read request is given priority.

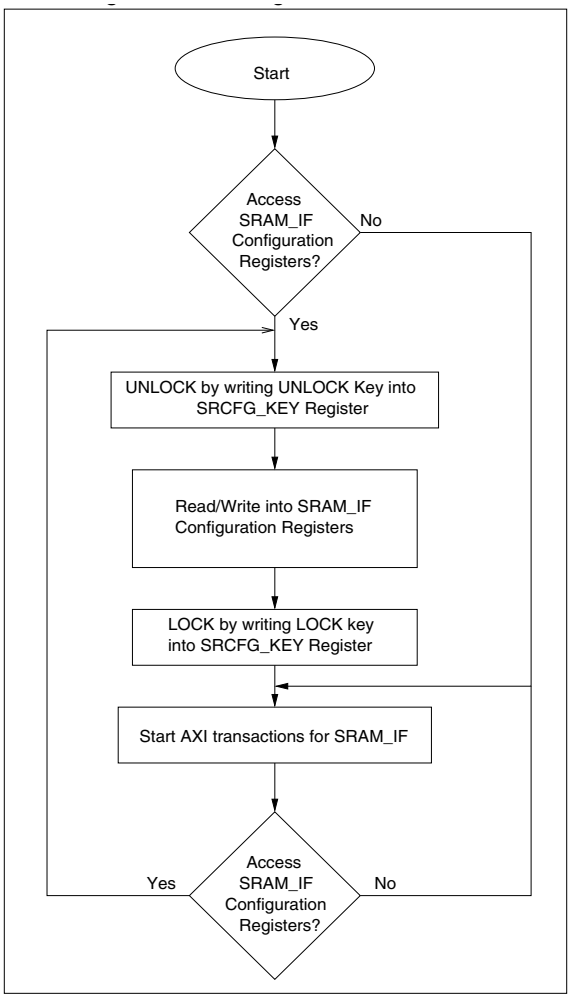
13.4 How to use the SRAM_IF

The following flowchart shows the sequence of steps to be followed to configure the SRAM_IF registers.

Typical SRAM_IF configuration flow

Figure 13-11 shows the steps to write to SRAM_IF Configuration Registers.

Figure 13-11. Configuration flowchart



14. Resource Input Configuration

This chapter explains the function and operation of the Resource Input Configuration module.

14.1 Outline of the Resource Input Configuration

This section describes the features and the block diagram of the Resource Input Configuration.

Features of the Resource Input Configuration

The Resource Input Configuration module controls the mapping of each input of a resource to several port pins or internal signals. It implements the two-stage multiplexing functionality for the input pin of the selected resource. First-stage multiplexing chooses between different ports, while the second stage multiplexing chooses between different internal signals and port.

The features of the Resource Input Configuration module are as follows:

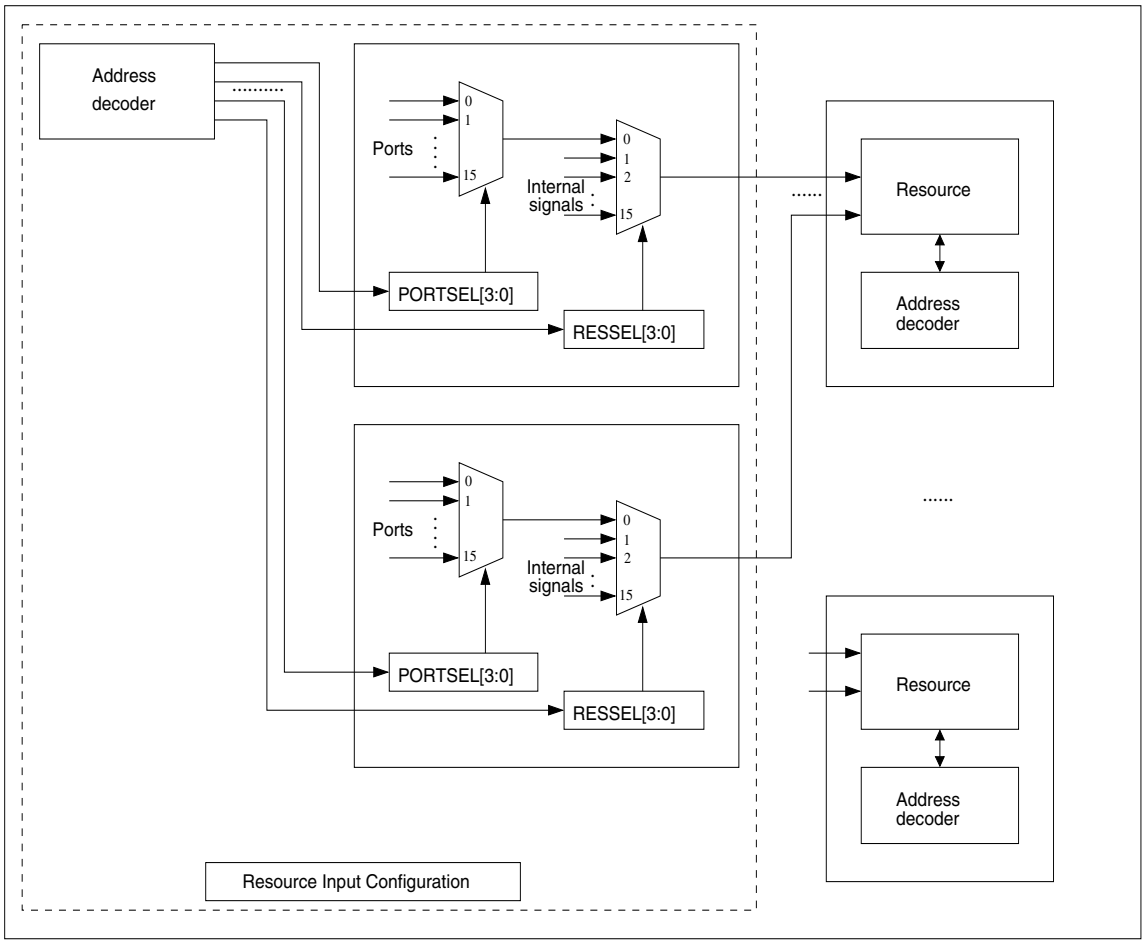
- A 16-bit Resource Input Configuration Register (RICFG_RESnIN) is associated with each input (IN) of a resource (RESn). Some examples include RICFG_CAN0RX, RICFG_FRT17TEXT etc.
- Selection between ports is performed by the RICFG_RESnIN:PORTSEL[3:0] register
- Selection between the internal signals and port is performed by the RICFG_RESnIN:RESSEL[3:0] register

Block diagram of the Resource Input Configuration

Figure 14-1 shows the Resource Input Configuration block diagram. The input of a resource can be driven either by an external ports or by internal signal driven by some other resource. Up to four port select bits (RICFG_RESnIN:PORTSEL[3:0]) associated with each resource input select between different ports. Up to four resource select bits (RICFG_RESnIN:RESSEL[3:0]) associated with each resource input select between internal signals or the port selected by port select bits.

Note: If any input of a resource is only internally connected and has no connection to a port then the port select bit does not exist for such resource input. Similarly if any input of a resource is only connected to a port and there is no internal signal connection, then the resource select bit does not exist for such resource input. If any input of a resource does not have any multiplexing the corresponding Resource Input Configuration Register does not exist.

Figure 14-1. Block diagram of Resource Input Configuration



14.2 Resource Input Configuration registers

This section describes the registers of the Resource Input Configuration.

Registers of the Resource Input Configuration

The following registers are available for the Resource Input Configuration:

Resource Input Configuration Registers (RICFG_RESnIN)

Memory layout of the Resource Input Configuration registers

Table 14-1. Memory layout of Resource Input Configuration registers

Offset	+1	+0
0x00000000 to 0x00007BFF	RICFG_RESnIN XXXX0000_XXXX0000	

14.2.1 Resource Input Configuration Registers (RICFG_RESnIN)

For each input of a peripheral resource a Resource Input Configuration Register (RICFG) exists near the resource and controls the input that can be mapped to several port pins (or internal signals).

Resource Input Configuration Registers (RICFG_RESnIN)

Figure 14-2. Resource Input Configuration Register (RICFG_RESnIN)

RICFG_RESnIN															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	PORTSEL[3]	PORTSEL[2]	PORTSEL[1]	PORTSEL[0]	reserved	reserved	reserved	reserved	RESSEL[3]	RESSEL[2]	RESSEL[1]	RESSEL[0]
-	-	-	-	RpWp	RpWp	RpWp	RpWp	-	-	-	-	RpWp	RpWp	RpWp	RpWp
X	X	X	X	0	0	0	0	X	X	X	X	0	0	0	0

Table 14-2. Resource Input Configuration Register (RICFG_RESnIN) bits

Bit Position	Bit Field Name	Bit Description
[15:12]	reserved	-

Table 14-2. Resource Input Configuration Register (RICFG_RESnIN) bits

Bit Position	Bit Field Name	Bit Description
[11:8]	PORTSEL	These bits select the source of the corresponding resource input. PORTSEL[3:0] = '0000': Source A PORTSEL[3:0] = '0001': Source B PORTSEL[3:0] = '0010': Source C PORTSEL[3:0] = '0011': Source D PORTSEL[3:0] = '0100': Source E PORTSEL[3:0] = '0101': Source F PORTSEL[3:0] = '0110: Source G PORTSEL[3:0] = '0111': Source H PORTSEL[3:0] = '1000': Source I PORTSEL[3:0] = '1001': Source J PORTSEL[3:0] = '1010': Source K PORTSEL[3:0] = '1011': Source L PORTSEL[3:0] = '1100: Source M PORTSEL[3:0] = '1101': Source N PORTSEL[3:0] = '1110': Source O PORTSEL[3:0] = '1111': Source P Sources A to P are configured as port pins or tied to logic zero. For resource inputs that can be mapped to no more than eight input pins, bit PORTSEL[3] is not implemented. For resource inputs that can be mapped to no more than four input pins, bits PORTSEL[3:2] are not implemented. For resource inputs that can be mapped to no more than eight two input pins, bits PORTSEL[3:1] are not implemented. For resource inputs that are mapped to a fixed pin, bits PORTSEL[3:0] are not implemented.
[7:4]	reserved	-

Table 14-2. Resource Input Configuration Register (RICFG_RESnIN) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	RESSEL	These bits select the source of the corresponding resource input. RESSEL[3:0] = '0000': Source A RESSEL[3:0] = '0001': Source B RESSEL[3:0] = '0010': Source C RESSEL[3:0] = '0011': Source D RESSEL[3:0] = '0100': Source E RESSEL[3:0] = '0101': Source F RESSEL[3:0] = '0110: Source G RESSEL[3:0] = '0111': Source H RESSEL[3:0] = '1000': Source I RESSEL[3:0] = '1001': Source J RESSEL[3:0] = '1010': Source K RESSEL[3:0] = '1011': Source L RESSEL[3:0] = '1100: Source M RESSEL[3:0] = '1101': Source N RESSEL[3:0] = '1110': Source O RESSEL[3:0] = '1111': Source P Source A is the output of the input pin muxing. Sources B to P are the outputs of other resources or are tied to logic zero. This can be used, for example, for interconnecting resources without using device pins. For resource inputs that can be mapped to no more than eight input pins, bit RESSEL[3] is not implemented. For resource inputs that can be mapped to no more than four input pins, bits RESSEL[3:2] are not implemented. For resource inputs that can be mapped to no more than eight two input pins, bits RESSEL[3:1] are not implemented. For resource inputs that are mapped to a fixed pin, bits RESSEL[3:0] are not implemented.

Note: There are some special registers which do not follow the above explanation; these registers are described in 14.4 Special connections

14.3 Notes on using the Resource Input Configuration

This section describes the register protection feature of the Resource Input Configuration.

Register protection feature

The Resource Input Configuration Register RICFG_RESnIN is protected in both user mode and privileged mode against read and write access; this is done separately by a global setting through a Peripheral Protection Unit (PPU) permission. Writing to this register either in user mode or privileged mode is permitted only if the global pin configuration access permission bit for the corresponding mode is set to '1'. Similarly, reading this register either in user mode or privileged mode is permitted only if the global pin configuration read permission bit or access permission bit for the respective mode is set to '1'.

All read and write permission violations are signalled to the Bus Error Collection Unit.

Table 14-3. Required permission for access to RICFG_RESnIN

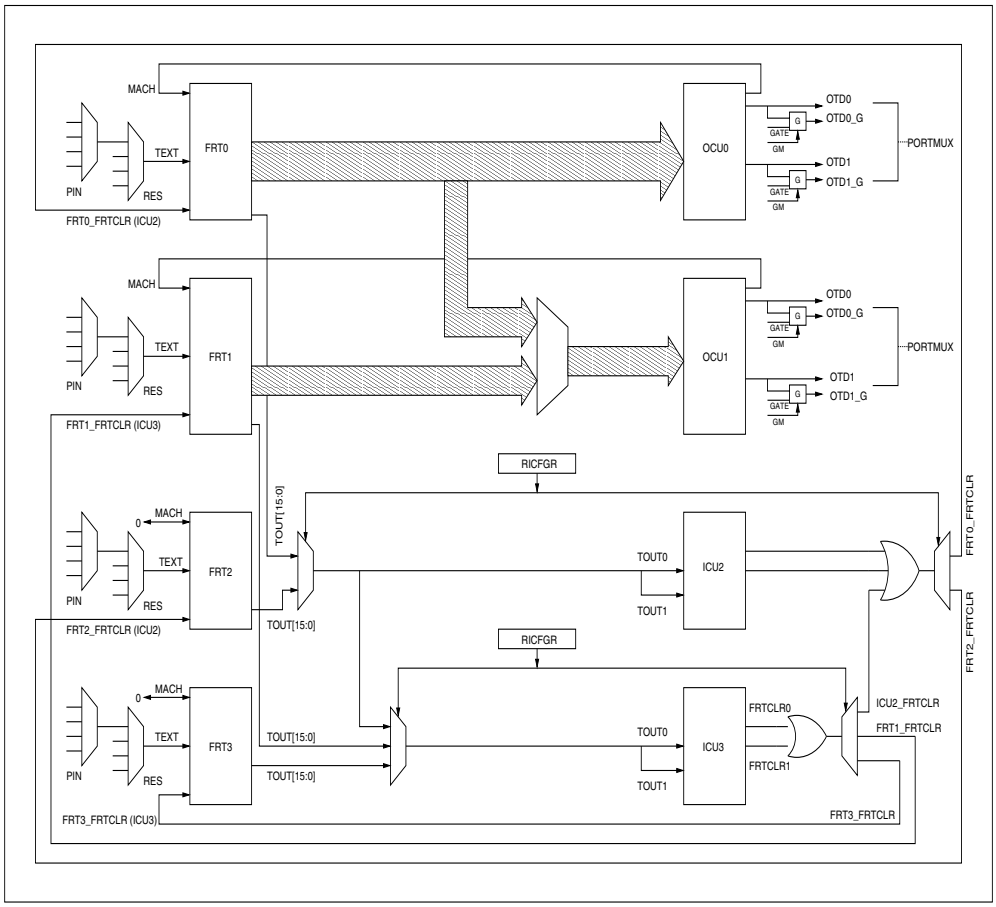
Access type	PPU read permission in user mode	PPU access permission in user mode	PPU read permission in privileged mode	PPU access permission in privileged mode
No access in user mode	'0'	'0'	X	X
Read and write accesses in user mode are not permitted	'1'	'0'	X	X
Read and write accesses in user mode are permitted	X	'1'	X	X
No access in privileged mode	X	X	'0'	'0'
Read and write access in pre-vileged mode are not permitted	X	X	'1'	'0'
Read access and write accesses in privileged mode are permirred	X	X	X	'1'

14.4 Special connections

This section includes figures for IO timer connections, the gated resource input configuration and ADC connections for ADC_EDGI, ADC_TIMI, and ADC_SMC.

IO timer connections

Figure 14-3. IO timer connections



Note: Similarly, the next set of FRT, OCU, ICU are connected together. For example FRT16-19, OCU16-17, ICU18-19 are connected together.

Resource gating configuration

The resource gating configuration provides the provision to gate a resource signal with other resource outputs to have the possibility to generate complex waveforms.

For example:

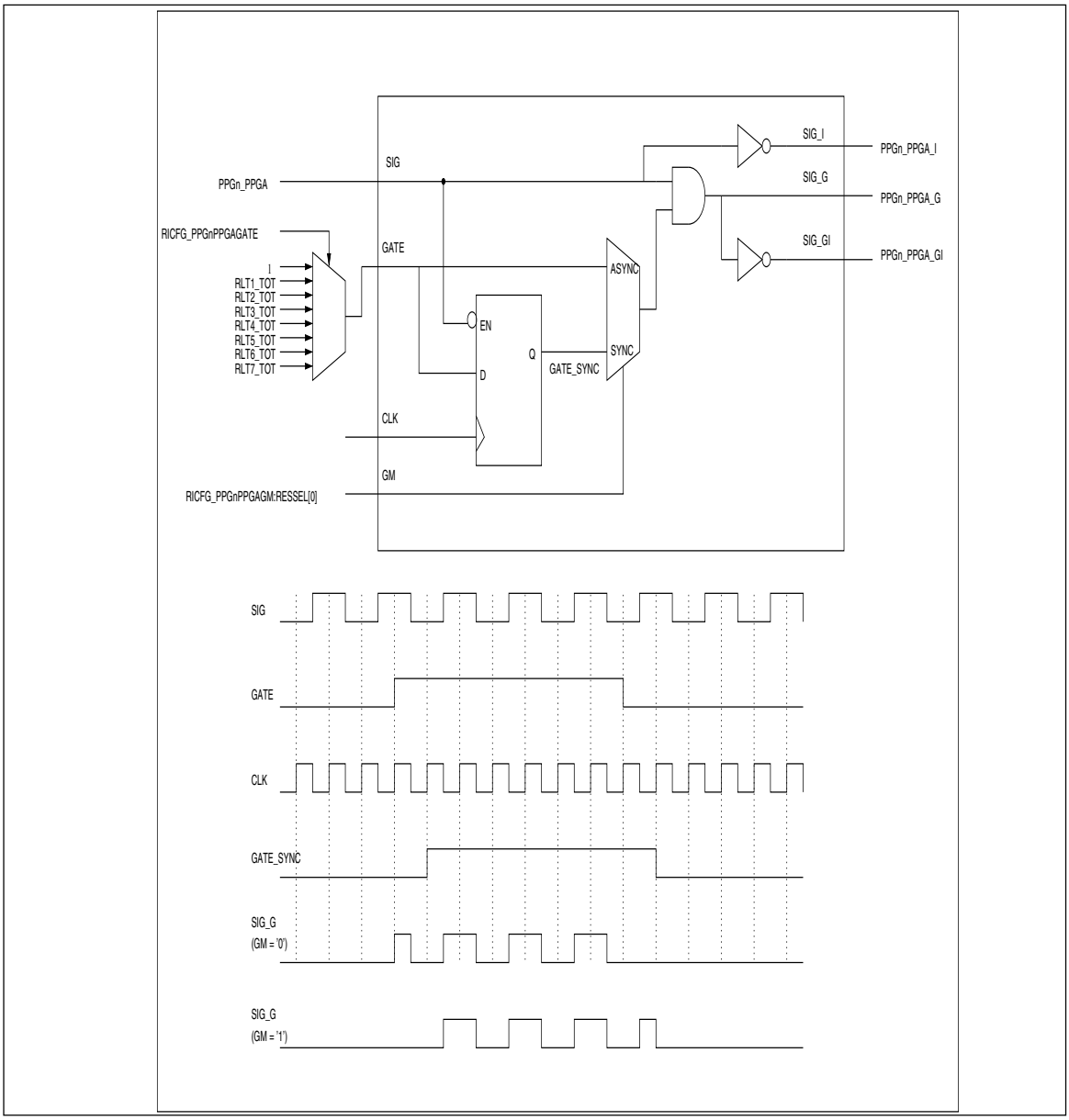
The PPG output (PPGn_PPGA) can be gated with different RLTs (RLTn_TOT). The gated signal (PPGn_PPGS_G) then goes to the device port or other peripherals.

The RICFG_PPGnPPGAGATE:RESSEL[3:0] register selects between different RLTn_TOT. The RICFG_PPGnP-PGAGM:RESSEL[3:0] register selects between two gating modes.

For more details refer to Figure 14-4.

Since these registers are resource specific, the complete list of registers are given in each device-specific datasheet.

Figure 14-4. Gated resource input configuration



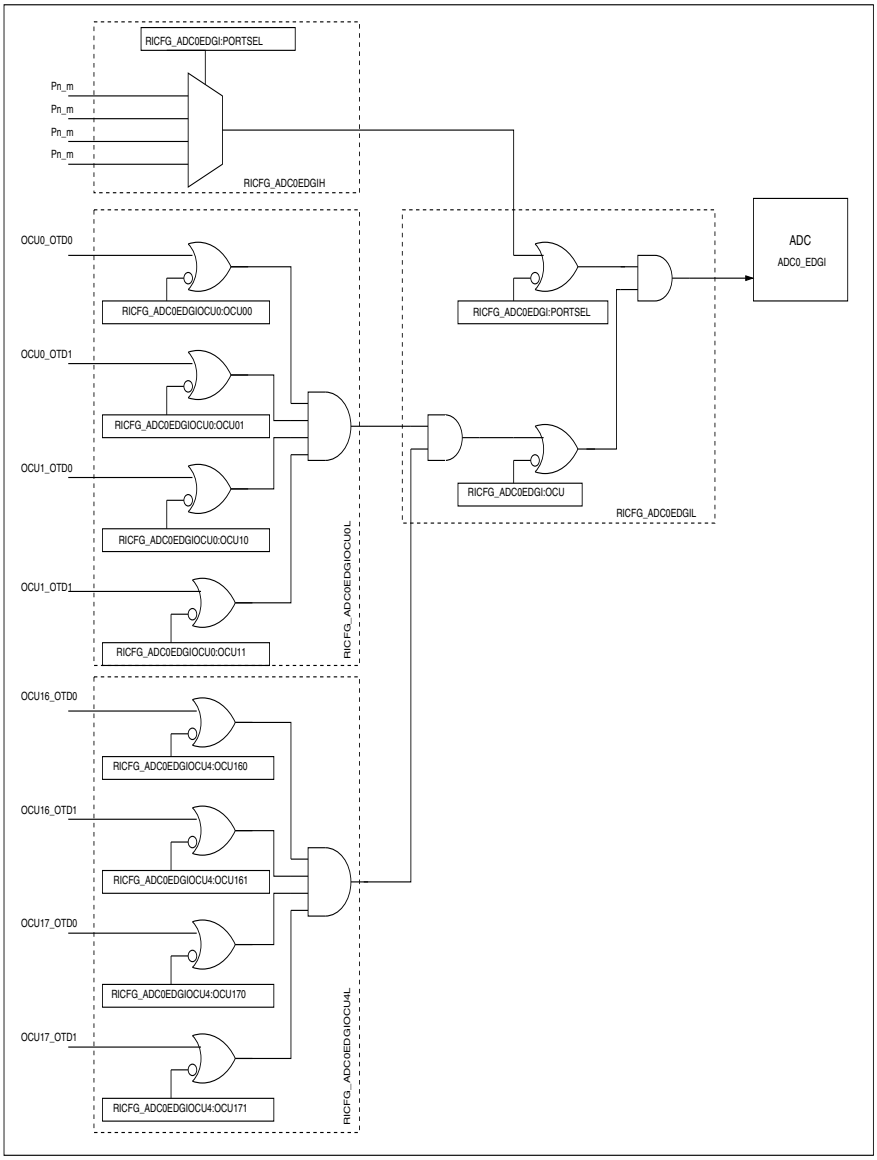
ADC connections

ADC_EDGI

Port selection for ADC0_EDGI is done by the RICFG_ADC0EDGI:PORTSEL register. Thge enabling of each the OCU outputs is done by the RICFG_ADC0EDGIOCUn register. The enabling of the port signal and OCU outputs is done by the RICFG_ADC0EDGI:PORTSEL and RICFG_ADC0EDGIOCUn signals respectively. Since the ADC0_EDGI signal is active low, the enabling/disabling of each individual signal is performed using an OR gate; the individual signals are then ANDed together to generate ADC0_EDGI.

For more details on these registers and for the actual port assignment refer to the device-specific datasheet.

Figure 14-5. Resource input configuration for ADC0_EDGI



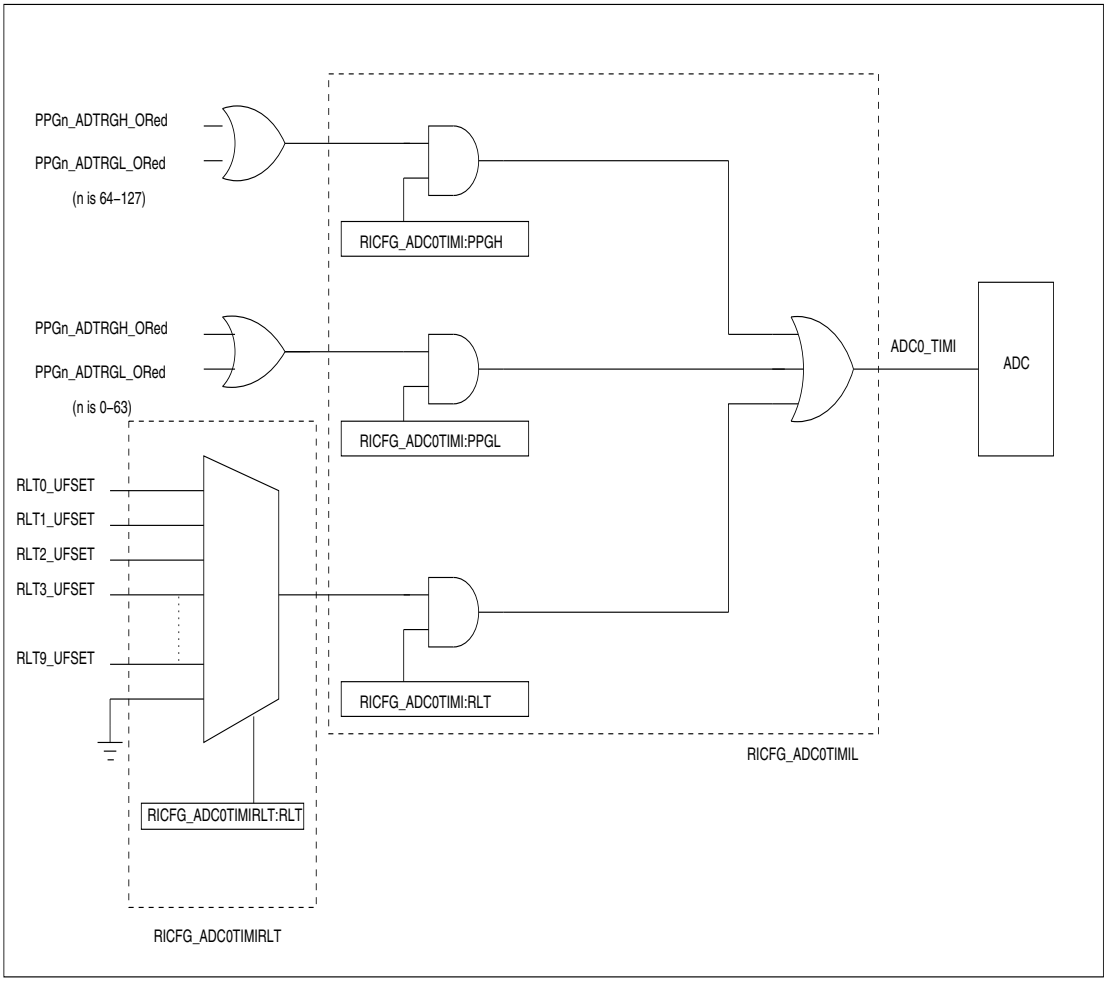
ADC_TIMI

The resource input ADC0_TIMI can be configured from the RLTn_UFSET, PPGn_ADTRGL and PPGn_ADTRGH signals. The selection of RLTn_UFSET is done by RICFG_ADC0TIMIRLT:RLT. The UFSET output is not configured to the resource input unless the mask bit RICFG_ADC0TIMI:RLT is set.

PPGn_ADTRGL and PPGn_ADTRGH (where n is 0-63) are ORed together which can be enabled by RICFG_ADC0TIMI:PPGL. Similarly PPGn_ADTRGL and PPGn_ADTRGH (where n is 64-127)are ORed together which can be enabled by RICFG_ADC0TIMI:PPGH.

For more details on these registers refer to the device-specific datasheet.

Figure 14-6. Resource input configuration for ADC0_TIMI



ADC_SMC connection

Figure 14-7 depicts an example for the resource input ADC analog input (ADC0_AN26). Device ports P1_00, P1_01, P1_02, and P1_03 are used as a source for ADC0_AN26 by setting RICFG_ADC0AN26:PORTSEL[3:0].

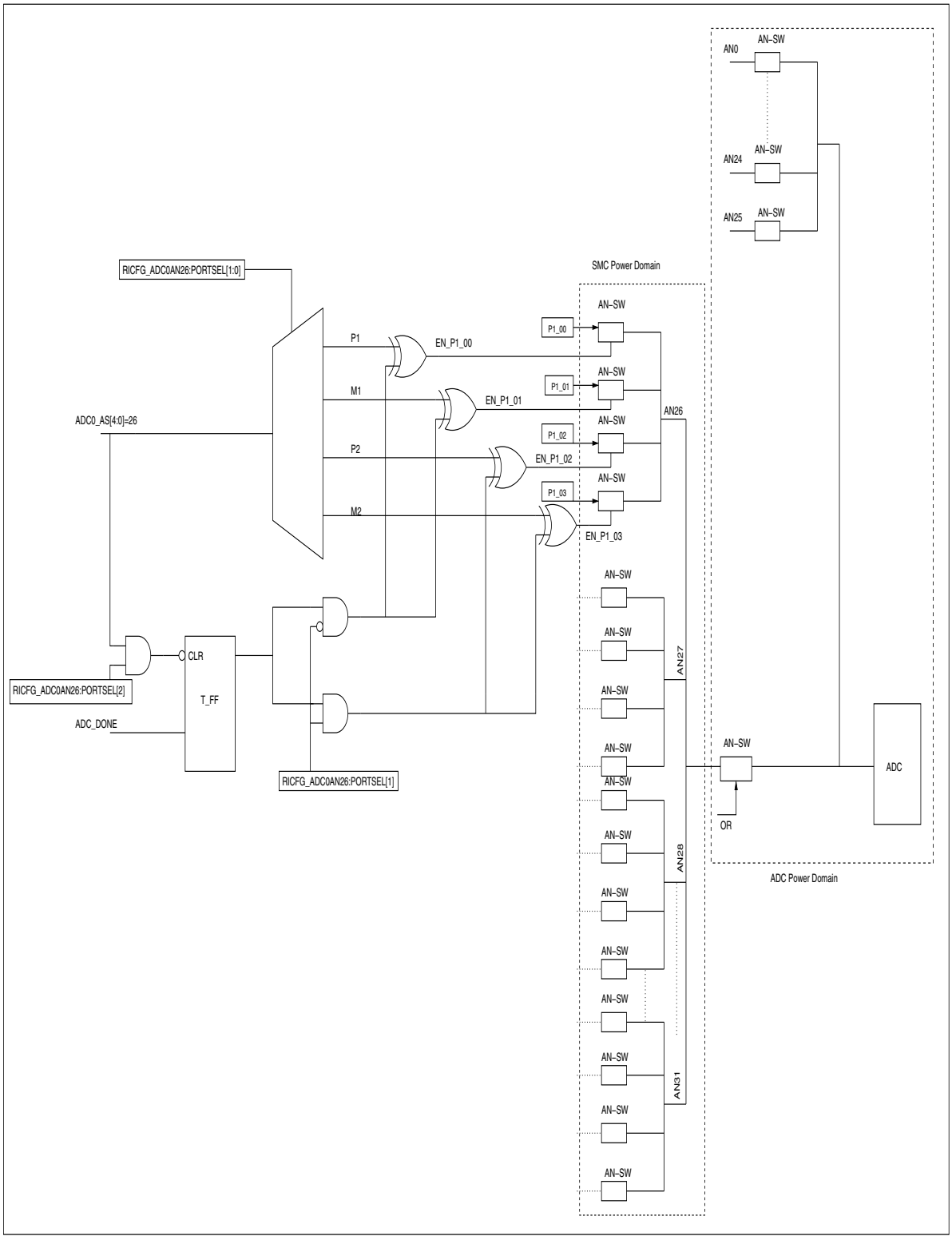
When RICFG_ADC0AN26:PORTSEL[3:0] is 0, then (P1_00) is selected as the source for ADC0_AN26. Similarly when RICFG_ADC0AN26:PORTSEL[3:0] is 3, then fourth pin (P1_03) is selected as the source.

When RICFG_ADC0AN26:PORTSEL[3:0] is set to 4, P1_00 and P1_01 are used as the source for ADC0_AN26 for every alternate conversion. Initially the port P1_00 is selected as the source for ADC0_AN26. Once conversion is done the ADC_DONE signal gets raised and the source input changes to port P1_01. On the next conversion P1_00 is selected followed by P1_01 and so on.

Similarly, when RICFG_ADC0AN26:PORTSEL[3:0] is set to 5, P1_01 and P1_00 are used as the source for ADC0_AN26 for every alternate conversion; when RICFG_ADC0AN26:PORTSEL[3:0] is set to 6, P1_02 and P1_03 are used; and when RICFG_ADC0AN26:PORTSEL[3:0] is set to 7, P1_03 and P1_02 are used.

For more details on these registers and for the actual port assignment refer the device-specific datasheet.

Figure 14-7. Resource input configuration for ADC_SMC



15. Boot ROM Hardware Interface

This chapter explains the functions and operations of the Boot ROM Hardware Interface (BootROM_IF).

15.1 Outline of Boot ROM Hardware Interface

The Boot ROM Hardware Interface is the hardware interface between the processor and the Boot ROM. This interface also implements the configuration of the exception vector register set. This section describes the features and the block diagram of the Boot ROM Hardware Interface.

Features of Boot ROM Hardware Interface

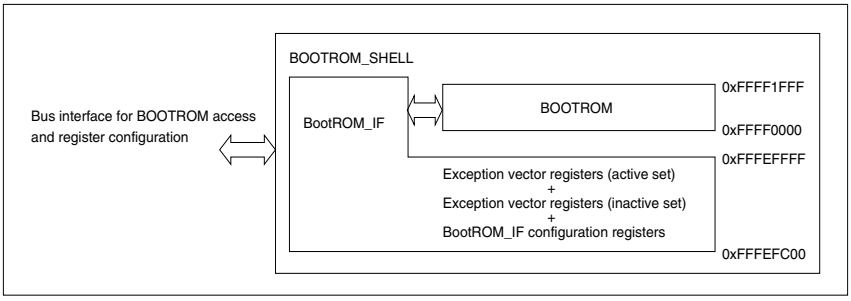
The Boot ROM Hardware Interface is the hardware interface between the processor and the Boot ROM. This section lists the features of Boot ROM Hardware Interface in detail:

- Configuration registers for user exception handlers (applicable only when Arm® Cortex® -R4's HIVECS option is '1')
- Two sets of exception vector registers, which enable the software to configure one set when another set is in use (applicable only when Arm® Cortex® -R4's HIVECS option is '1')
- Error generation if write access to locked configuration registers occurs
- Error generation if write to the ROM area occurs. This area is treated as read-only region

Block diagram of Boot ROM Hardware Interface

The Boot ROM is mapped at 0xFFFF0000 (Arm® Cortex® -R4 high exception vectors).

Figure 15-1. Block diagram of Boot ROM Hardware Interface



15.2 Boot ROM Hardware Interface registers

All registers in the Boot ROM Hardware Interface module are explained in this section.

Registers of Boot ROM Hardware Interface

The following registers are available for Boot ROM Hardware Interface:

- Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK)
- Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG)
- Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT)
- Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT)
- Prefetch Abort Vector Register - Inactive Set (EXCFG_PABORTINACT)
- Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT)
- Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT)
- Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT)
- Supervisor Call Vector Register - Active Set (EXCFG_SVCACT)
- Prefetch Abort Vector Register - Active Set (EXCFG_PABORTACT)
- Data Abort Vector Register - Active Set (EXCFG_DABORTACT)
- Interrupt Vector Register - Active Set (EXCFG_IRQACT)

Memory layout of Boot ROM Hardware Interface register

Table 15-1. Memory layout of Boot ROM Hardware Interface register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000 - 0x00000350	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000358	read0 00000000 00000000 00000000 00000000				EXCFG_UNLOCK 00000000 00000000 00000000 00000000			
0x00000360	read0 00000000 00000000 00000000 00000000				EXCFG_CNFG 00000000 00000000 00000000 00000001			
0x00000368 - 0x00000378	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000380	EXCFG_UNDEFINACT 11111111 11111111 00000000 00100100				read0 00000000 00000000 00000000 00000000			
0x00000388	EXCFG_PABORTINACT 11111111 11111111 00000000 00101100				EXCFG_SVCINACT 11111111 11111111 00000000 00101000			
0x00000390	read0 00000000 00000000 00000000 00000000				EXCFG_DABORTINACT 11111111 11111111 00000000 00110000			
0x00000398	read0 00000000 00000000 00000000 00000000				EXCFG_IRQINACT 11111111 11111111 00000000 00111000			
0x000003A0 - 0x000003B8	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							

Table 15-1. Memory layout of Boot ROM Hardware Interface register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000003C0	EXCFG_UNDEFACT 11111111 11111111 00000000 00100100				read0 00000000 00000000 00000000 00000000			
0x000003C8	EXCFG_PABORTACT 11111111 11111111 00000000 00101100				EXCFG_SVCACT 11111111 11111111 00000000 00101000			
0x000003D0	read0 00000000 00000000 00000000 00000000				EXCFG_DABORTACT 11111111 11111111 00000000 00110000			
0x000003D8	read0 00000000 00000000 00000000 00000000				EXCFG_IRQACT 11111111 11111111 00000000 00111000			

15.2.1 Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK)

This register decides the write accessibility of the configuration registers of this module. It has to be written with the specific unlock value (0xACC5B007) to reset the EXCFG_CNFG:LOCKSTATUS bit, which allows privileged write to the configuration registers. After configuration, this register should be written with the specific lock value (0xB007ECF6) which sets the EXCFG_CNFG:LOCKSTATUS bit and restricts the write access to the configuration registers. This register can be written only with 32/64-bit privileged access.

Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK)

Figure 15-2. Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK)

EXCFG_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-2. Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	BootROM_IF Unlock This unlock register protects the BootROM_IF registers from being modified accidentally by software. The configuration and exception vector registers cannot be written until this register has been written with a specific (unlock) value. This register only supports 32/64 bits privileged access. The read to this register always returns '0'. To lock the BootROM_IF registers again, software must write another value specific to lock. Access to configuration registers when locked, or writing a value other than lock or unlock value to this register, or any write other than 32/64-bit privileged write access to this register causes protection error.

15.2.2 Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG)

This register contains a configuration bit and a status bit. EXCFG_CNFG:LOCKSTATUS (status bit) shows the lock status of the configuration registers. By default this bit is '1', i.e. all the configuration registers are locked for write access. To unlock or to reset this bit, specific unlock value has to be written to EXCFG_UNLOCK register. This bit can again be set by writing the lock value in the EXCFG_UNLOCK register. EXCFG_CNFG:SWAPREG (configuration bit) is used to swap the contents of the active and the inactive set of exception vector registers. This register is only writable in privileged mode and when the EXCFG_CNFG:LOCKSTATUS = '0'.

Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG)

Figure 15-3. Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG)

EXCFG_CNFG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SWAPREG	read0	read0	read0	read0	read0	read0	read0	LOCKSTATUS
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 15-3. Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-

Table 15-3. Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	SWAPREG	Swap Exception Register This bit enables the user to select between the two available configuration sets (active/inactive): '0': No effect '1': Triggers swapping of the contents of active and inactive register set Read always returns '0'.
[7:1]	read0	-
[0]	LOCKSTATUS	BootROM_IF Lock Status This bit provides locked/unlocked status of BootROM_IF configuration registers. '0': Configuration registers are unlocked for write access '1': Configuration registers are locked for write access By default this bit is '1', writing the correct unlock value to EXCFG_UNLOCK register resets this bit to '0' and writing the correct lock value sets this bit to '1'.

Note: There should be a time window of 30 to 40 CPU clock cycles, after writing '1' to the EXCFG_CNFG:SWAPREG bit, where both exception handlers must be valid. It is not deterministic which handler is executed during this time.

15.2.3 Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT)

This register contains the start address of the exception handler for undefined instruction exception. It is a part of inactive set exception vector register. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT) are swapped. This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT)

Figure 15-4. Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT)

EXCFG_UNDEFINACT																															
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**
UNDEFVEC[31]	UNDEFVEC[30]	UNDEFVEC[29]	UNDEFVEC[28]	UNDEFVEC[27]	UNDEFVEC[26]	UNDEFVEC[25]	UNDEFVEC[24]	UNDEFVEC[23]	UNDEFVEC[22]	UNDEFVEC[21]	UNDEFVEC[20]	UNDEFVEC[19]	UNDEFVEC[18]	UNDEFVEC[17]	UNDEFVEC[16]	UNDEFVEC[15]	UNDEFVEC[14]	UNDEFVEC[13]	UNDEFVEC[12]	UNDEFVEC[11]	UNDEFVEC[10]	UNDEFVEC[9]	UNDEFVEC[8]	UNDEFVEC[7]	UNDEFVEC[6]	UNDEFVEC[5]	UNDEFVEC[4]	UNDEFVEC[3]	UNDEFVEC[2]	UNDEFVEC[1]	UNDEFVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Table 15-4. Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNDEFVEC	Undefined Instruction Vector This register contains the start address of the exception handler for undefined instruction exception. It is an inactive set register.

15.2.4 Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT)

This register contains the start address of the exception handler for supervisor call exception. It is a part of inactive set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Supervisor Call Vector Register - Active Set (EXCFG_SVCACT) are swapped.

This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT)

Figure 15-5. Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT)

EXCFG_SVCINACT																															
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**
SVCVEC[31]	SVCVEC[30]	SVCVEC[29]	SVCVEC[28]	SVCVEC[27]	SVCVEC[26]	SVCVEC[25]	SVCVEC[24]	SVCVEC[23]	SVCVEC[22]	SVCVEC[21]	SVCVEC[20]	SVCVEC[19]	SVCVEC[18]	SVCVEC[17]	SVCVEC[16]	SVCVEC[15]	SVCVEC[14]	SVCVEC[13]	SVCVEC[12]	SVCVEC[11]	SVCVEC[10]	SVCVEC[9]	SVCVEC[8]	SVCVEC[7]	SVCVEC[6]	SVCVEC[5]	SVCVEC[4]	SVCVEC[3]	SVCVEC[2]	SVCVEC[1]	SVCVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0

Table 15-5. Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SVCVEC	Supervisor Call Vector This register contains the start address of the exception handler for supervisor call exception. It is an inactive set register.

15.2.5 Prefetch Abort Vector Register - Inactive Set (EXCFG_PABORTINACT)

This register contains the start address of the exception handler for prefetch abort exception. It is a part of inactive set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Prefetch Abort Vector Register - Active Set (EXCFG_ PABORTACT) are swapped. This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Prefetch Abort Vector Register - Inactive Set (EXCFG_PABORTINACT)

Figure 15-6. Prefetch Abort Vector Register - Inactive Set (EXCFG_PABORTINACT)

EXCFG_PABORTINACT																															
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**
PABORTVEC[31]	PABORTVEC[30]	PABORTVEC[29]	PABORTVEC[28]	PABORTVEC[27]	PABORTVEC[26]	PABORTVEC[25]	PABORTVEC[24]	PABORTVEC[23]	PABORTVEC[22]	PABORTVEC[21]	PABORTVEC[20]	PABORTVEC[19]	PABORTVEC[18]	PABORTVEC[17]	PABORTVEC[16]	PABORTVEC[15]	PABORTVEC[14]	PABORTVEC[13]	PABORTVEC[12]	PABORTVEC[11]	PABORTVEC[10]	PABORTVEC[9]	PABORTVEC[8]	PABORTVEC[7]	PABORTVEC[6]	PABORTVEC[5]	PABORTVEC[4]	PABORTVEC[3]	PABORTVEC[2]	PABORTVEC[1]	PABORTVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

Table 15-6. Prefetch Abort Vector Register - Inactive Set (EXCFG_PABORTINACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PABORTVEC	Prefetch Abort Vector This register contains the start address of the exception handler for prefetch abort exception. It is an inactive set register.

15.2.6 Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT)

This register contains the start address of the exception handler for data abort exception. It is a part of inactive set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Data Abort Vector Register - Active Set (EXCFG_DABORTACT) are swapped.

This register is used only when Arm® Cortex® -R4’sHIVECS option is '1'.

Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT)

Figure 15-7. Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT)

EXCFG_DABORTINACT																															
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**
DABORTVEC[31]	DABORTVEC[30]	DABORTVEC[29]	DABORTVEC[28]	DABORTVEC[27]	DABORTVEC[26]	DABORTVEC[25]	DABORTVEC[24]	DABORTVEC[23]	DABORTVEC[22]	DABORTVEC[21]	DABORTVEC[20]	DABORTVEC[19]	DABORTVEC[18]	DABORTVEC[17]	DABORTVEC[16]	DABORTVEC[15]	DABORTVEC[14]	DABORTVEC[13]	DABORTVEC[12]	DABORTVEC[11]	DABORTVEC[10]	DABORTVEC[9]	DABORTVEC[8]	DABORTVEC[7]	DABORTVEC[6]	DABORTVEC[5]	DABORTVEC[4]	DABORTVEC[3]	DABORTVEC[2]	DABORTVEC[1]	DABORTVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 15-7. Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DABORTVEC	Data Abort Vector This register contains the start address of the exception handler for data abort exception. It is an inactive set register.

15.2.7 Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT)

This register contains the start address of the exception handler for interrupt exception if the VIC port is not used. It is a part of inactive set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Interrupt Vector Register - Active Set (EXCFG_IRQACT) are swapped.

This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT)

Figure 15-8. Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT)

EXCFG_IRQINACT																															
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**
IRQVEC[31]	IRQVEC[30]	IRQVEC[29]	IRQVEC[28]	IRQVEC[27]	IRQVEC[26]	IRQVEC[25]	IRQVEC[24]	IRQVEC[23]	IRQVEC[22]	IRQVEC[21]	IRQVEC[20]	IRQVEC[19]	IRQVEC[18]	IRQVEC[17]	IRQVEC[16]	IRQVEC[15]	IRQVEC[14]	IRQVEC[13]	IRQVEC[12]	IRQVEC[11]	IRQVEC[10]	IRQVEC[9]	IRQVEC[8]	IRQVEC[7]	IRQVEC[6]	IRQVEC[5]	IRQVEC[4]	IRQVEC[3]	IRQVEC[2]	IRQVEC[1]	IRQVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Table 15-8. Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQVEC	Interrupt Vector This register contains the start address of the exception handler for interrupt exception. It is an inactive set register.

Note:

If VIC port is used, the interrupt vector provided from the VIC port can be directly used and this register need not be used. This register will hold the default value of the interrupt vector if software has not overwritten with a new value.

If VIC port is not used, this register stores the interrupt vector as programmed by the software.

For VIC port description, refer to Arm® Technical Reference Manual.

15.2.8 Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT)

This register contains the start address of the exception handler for undefined instruction exception. It is a part of active set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Undefined Instruction Vector Register - Inactive Set (EXCFG_UNDEFINACT) are swapped. This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT)

Figure 15-9. Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT)

EXCFG_UNDEFACT																															
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**
UNDEFVEC[31]	UNDEFVEC[30]	UNDEFVEC[29]	UNDEFVEC[28]	UNDEFVEC[27]	UNDEFVEC[26]	UNDEFVEC[25]	UNDEFVEC[24]	UNDEFVEC[23]	UNDEFVEC[22]	UNDEFVEC[21]	UNDEFVEC[20]	UNDEFVEC[19]	UNDEFVEC[18]	UNDEFVEC[17]	UNDEFVEC[16]	UNDEFVEC[15]	UNDEFVEC[14]	UNDEFVEC[13]	UNDEFVEC[12]	UNDEFVEC[11]	UNDEFVEC[10]	UNDEFVEC[9]	UNDEFVEC[8]	UNDEFVEC[7]	UNDEFVEC[6]	UNDEFVEC[5]	UNDEFVEC[4]	UNDEFVEC[3]	UNDEFVEC[2]	UNDEFVEC[1]	UNDEFVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

Table 15-9. Undefined Instruction Vector Register - Active Set (EXCFG_UNDEFACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNDEFVEC	Undefined Instruction Vector This register contains the start address of the exception handler for undefined instruction exception. It is an active set register.

15.2.9 Supervisor Call Vector Register - Active Set (EXCFG_SVCACT)

This register contains the start address of the exception handler for supervisor call exception. It is a part of active set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Supervisor Call Vector Register - Inactive Set (EXCFG_SVCINACT) are swapped.

This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Supervisor Call Vector Register - Active Set (EXCFG_SVCACT)

Figure 15-10. Supervisor Call Vector Register - Active Set (EXCFG_SVCACT)

EXCFG_SVCACT																															
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**
SVCVEC[31]	SVCVEC[30]	SVCVEC[29]	SVCVEC[28]	SVCVEC[27]	SVCVEC[26]	SVCVEC[25]	SVCVEC[24]	SVCVEC[23]	SVCVEC[22]	SVCVEC[21]	SVCVEC[20]	SVCVEC[19]	SVCVEC[18]	SVCVEC[17]	SVCVEC[16]	SVCVEC[15]	SVCVEC[14]	SVCVEC[13]	SVCVEC[12]	SVCVEC[11]	SVCVEC[10]	SVCVEC[9]	SVCVEC[8]	SVCVEC[7]	SVCVEC[6]	SVCVEC[5]	SVCVEC[4]	SVCVEC[3]	SVCVEC[2]	SVCVEC[1]	SVCVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0

Table 15-10. Supervisor Call Vector Register - Active Set (EXCFG_SVCACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SVCVEC	Supervisor Call Vector This register contains the start address of the exception handler for supervisor call exception It is an active set register.

15.2.10 Prefetch Abort Vector Register - Active Set (EXCFG_PABORTACT)

This register contains the start address of the exception handler for prefetch abort exception. It is a part of active set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Prefetch Abort Vector Register - Inactive Set (EXCFG_ PABORTINACT) are swapped.

This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Prefetch Abort Vector Register - Active Set (EXCFG_PABORTACT)

Figure 15-11. Prefetch Abort Vector Register - Active Set (EXCFG_PABORTACT)

EXCFG_PABORTACT																															
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**
PABORTVEC[31]	PABORTVEC[30]	PABORTVEC[29]	PABORTVEC[28]	PABORTVEC[27]	PABORTVEC[26]	PABORTVEC[25]	PABORTVEC[24]	PABORTVEC[23]	PABORTVEC[22]	PABORTVEC[21]	PABORTVEC[20]	PABORTVEC[19]	PABORTVEC[18]	PABORTVEC[17]	PABORTVEC[16]	PABORTVEC[15]	PABORTVEC[14]	PABORTVEC[13]	PABORTVEC[12]	PABORTVEC[11]	PABORTVEC[10]	PABORTVEC[9]	PABORTVEC[8]	PABORTVEC[7]	PABORTVEC[6]	PABORTVEC[5]	PABORTVEC[4]	PABORTVEC[3]	PABORTVEC[2]	PABORTVEC[1]	PABORTVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

Table 15-11. Prefetch Abort Vector Register - Active Set (EXCFG_PABORTACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PABORTVEC	Prefetch Abort Vector This register contains the start address of the exception handler for prefetch abort exception. It is an active set register.

15.2.11 Data Abort Vector Register - Active Set (EXCFG_DABORTACT)

This register contains the start address of the exception handler for data abort exception. It is a part of active set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCKSTATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Data Abort Vector Register - Inactive Set (EXCFG_DABORTINACT) are swapped.

This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Data Abort Vector Register - Active Set (EXCFG_DABORTACT)

Figure 15-12. Data Abort Vector Register - Active Set (EXCFG_DABORTACT)

EXCFG_DABORTACT																															
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**
DABORTVEC[31]	DABORTVEC[30]	DABORTVEC[29]	DABORTVEC[28]	DABORTVEC[27]	DABORTVEC[26]	DABORTVEC[25]	DABORTVEC[24]	DABORTVEC[23]	DABORTVEC[22]	DABORTVEC[21]	DABORTVEC[20]	DABORTVEC[19]	DABORTVEC[18]	DABORTVEC[17]	DABORTVEC[16]	DABORTVEC[15]	DABORTVEC[14]	DABORTVEC[13]	DABORTVEC[12]	DABORTVEC[11]	DABORTVEC[10]	DABORTVEC[9]	DABORTVEC[8]	DABORTVEC[7]	DABORTVEC[6]	DABORTVEC[5]	DABORTVEC[4]	DABORTVEC[3]	DABORTVEC[2]	DABORTVEC[1]	DABORTVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 15-12. Data Abort Vector Register - Active Set (EXCFG_DABORTACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DABORTVEC	Data Abort Vector This register contains the start address of the exception handler for data abort exception. It is an active set register.

15.2.12 Interrupt Vector Register - Active Set (EXCFG_IRQACT)

This register contains the start address of the exception handler for interrupt exception if VIC port is not used. It is a part of active set exception vector registers. This register can be written only in privileged mode and when EXCFG_CNFG:LOCK-STATUS = '0'. When '1' is written to EXCFG_CNFG:SWAPREG, the contents of this register and the Interrupt Vector Register - Inactive Set (EXCFG_IRQINACT) are swapped.

This register is used only when Arm® Cortex® -R4's HIVECS option is '1'.

Interrupt Vector Register - Active Set (EXCFG_IRQACT)

Figure 15-13. Interrupt Vector Register - Active Set (EXCFG_IRQACT)

EXCFG_IRQACT																															
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**
IRQVEC[31]	IRQVEC[30]	IRQVEC[29]	IRQVEC[28]	IRQVEC[27]	IRQVEC[26]	IRQVEC[25]	IRQVEC[24]	IRQVEC[23]	IRQVEC[22]	IRQVEC[21]	IRQVEC[20]	IRQVEC[19]	IRQVEC[18]	IRQVEC[17]	IRQVEC[16]	IRQVEC[15]	IRQVEC[14]	IRQVEC[13]	IRQVEC[12]	IRQVEC[11]	IRQVEC[10]	IRQVEC[9]	IRQVEC[8]	IRQVEC[7]	IRQVEC[6]	IRQVEC[5]	IRQVEC[4]	IRQVEC[3]	IRQVEC[2]	IRQVEC[1]	IRQVEC[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Figure 15-14. Interrupt Vector Register - Active Set (EXCFG_IRQACT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQVEC	Interrupt Vector This register contains the start address of the exception handler for interrupt exception. It is an active set register.

Note:

This register contains the interrupt vector only when VIC port is not used. If VIC port is used for providing vector address, this register address space will be reserved.

For VIC port description, refer to Arm® Technical Reference Manual.

15.3 Operation of Boot ROM Hardware Interface

This chapter describes the operation of Boot ROM Hardware Interface.

Arm® Cortex-R4 exception vector handling

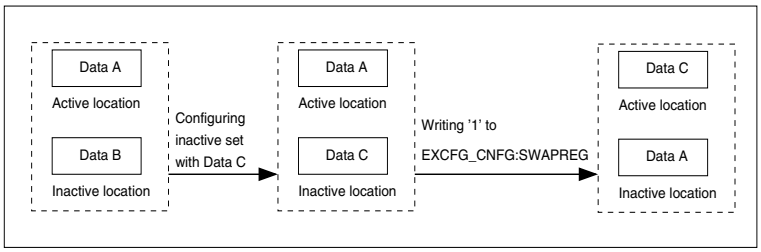
The exceptions in Arm® Cortex® - R4's are handled using the following steps:

1. The exception occurs.
2. The code jumps to the corresponding entry in the fixed exception vector table in the Boot ROM.
3. In the exception entry location, a load instruction exists, which loads the value of the respective exception vector register in the PC.
4. Thus a jump occurs to the start address of exception handler and the exception handler is executed.

To have a variable jump destination despite the fixed exception entry to vector table in the Boot ROM, there exist registers to configure the start addresses (inline literals) of the Arm(R) Cortex(R)-R4 exception handlers. These registers are mapped in the address locations below the Boot ROM, which is referred by the exception entry in the Boot ROM upon an exception. This enables the dynamic redefinition of the exception vectors.

To have the possibility to redefine all exception vector registers with one access there exist two sets of registers, active set and inactive set. Both the sets are mapped in the address locations below the Boot ROM as shown in Table 15-1. The address location to which the active set is mapped is the active location, which is referred by the exception entry in the Boot ROM upon an exception. The address location to which inactive set is mapped is the inactive location, which can be reconfigured by the software. The active set is also configurable. To exchange the sets, EXCFG_CNFG:SWAPREG bit (for more details refer to 15.2.2 Boot ROM Hardware Interface Configuration Register (EXCFG_CNFG)) is available.

Figure 15-15. Swapping of active and inactive sets



Notes:

1. There should be a time window of 30 to 40 CPU clock cycles, after writing '1' to the EXCFG_CNFG:SWAPREG bit, where both exception handlers must be valid. It is not deterministic which handler is executed during this time.
2. If the exception vector table is located at 0x00000000 by setting Arm® Cortex® -R4's HIVECS = '0', all exception handlers other than 'reset' have to be defined by a vector table in RAM.

Error generation

Boot ROM Hardware Interface generates an error, which causes a bus error response, when any of the following condition occurs:

- Writing a value different from lock or unlock to the EXCFG_UNLOCK register (for more details refer to 15.2.1 Boot ROM Hardware Interface Unlock Register (EXCFG_UNLOCK))
- Write access to the EXCFG_UNLOCK register other than with 32/64-bit
- Write access to the Boot ROM Hardware Interface configuration registers before unlocking them
- Non-privileged write access to the Boot ROM Hardware Interface registers
- Access to an unimplemented address (reserved) of the module
- Write access to the Boot ROM
- There is one part of the Boot ROM software which is not accessible. This section is for internal use by Cypress. Any access to this area will generate an error. Refer to device datasheet for the address range of this area

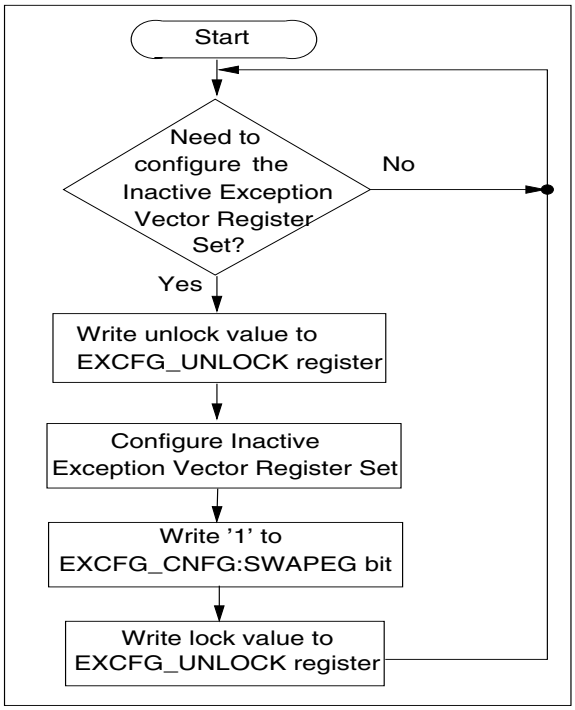
- Write access to a read-only byte

15.4 Boot ROM Hardware Interface flowchart

The following flowchart shows the sequence of steps to use the Boot ROM Hardware Interface.

Boot ROM Hardware Interface usage flowchart

Figure 15-16. Boot ROM Hardware Interface usage flowchart



16. Boot ROM Software Interface

This chapter explains the functions and operations of the Boot ROM Software Interface.

16.1 Outline of the Boot ROM Software Interface

The Boot ROM is the fixed built-in firmware, which is executed after each reset and before the user application. It controls Security Checker settings, the Security Bridge and start of application. It also supports debugging of application software and testing of the MCU. This chapter describes the functionality of Boot ROM and how to configure it by Security Description Record (SDR) and Boot Description Record (BDR) markers in Flash memory.

Features of the BOOT ROM

The Boot ROM provides the following features to start user application.

- Pin-configured Main mode is detected.
- User configured device start scheme is read from the Boot Description Record (BDR) in the Instruction Flash.
- A fixed Arm Cortex-R4 exception table is implemented at address FFFF:0000. User configurable exception handlers are referenced via the Interrupt Controller and Boot ROM Hardware Interface.
- Default exception handlers reset the device in the event an exception occurs before the user exception handler is configured.
- User code is started at configurable addresses, thereby enabling the safe update of application code and boot loader.
- The start of application is delayed until the debugger is connected. Optionally, the application is started without delay.

The Boot ROM provides the following features for device security to protect customer code against read-out in all device modes.

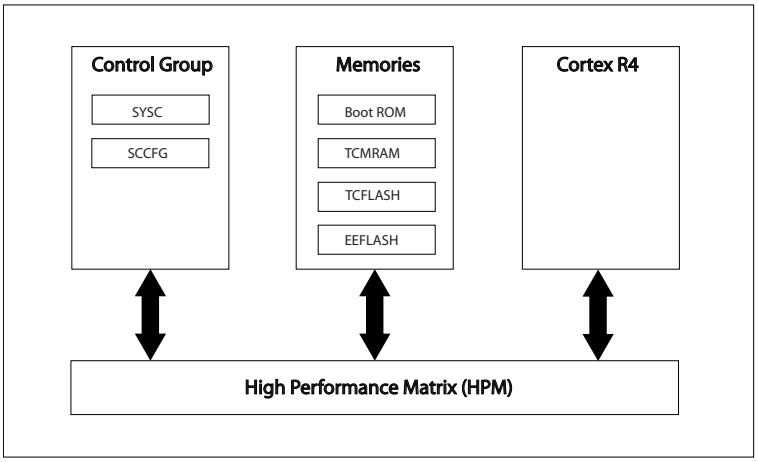
- The user configured security scheme is read from Security Description Records (SDRs) in the Instruction Flash and Data Flash.
- Device mode is detected and the user configured security scheme is applied to the Security Checker and System Controller.
- The user configured security scheme may use an alternative SDR in Flash enabling the secure update of application code and boot loader.
- In the event of suspicious security manipulation during boot phase, either the device is reset or restrictive setting is applied.

The Boot ROM provides the following features for the Secure Hardware Extension (SHE) if it is available on the device.

- Read Secure Boot configuration from BDR and request Secure Boot to SHECO with each reset.
- Optionally, wait for the Secure Boot to finish before starting the application (depending on the secure boot configuration).

Block Diagram of Boot ROM

Figure 16-1. Boot ROM Block diagram



16.2 Boot ROM markers in Flash memory

This section describes all Boot ROM markers in Flash memory. Markers are similar to registers because their value affects the behaviour of the built-in IP. However, unlike registers, the markers are not connected to specific IPs but their value is read and processed by Boot ROM firmware. Markers are grouped in records, which are allocated to different Flash sectors.

Boot ROM Markers

The following markers are available for Boot ROM.

- Boot Description Record (BDR):
  - SHE Secure Boot Mode Marker (BDR_SBMM)
  - SHE Secure Boot Size Marker (BDR_SBSM)
  - Debugger Waiting Enable Marker (BDR_DWEM)
  - Alternative Boot Vector Marker (BDR_ABVM)
  - Alternative Boot Vector Enable Marker (BDR_ABVEM)
- Main Security Description Record (MSDR):
  - Primary Device Security Enable Marker (MSDR_PDSEM)/Secondary Device Security Enable Marker (MSDR_SDSEM)
  - Primary Device Security Key Marker (MSDR_PDSKM0~3)/Secondary Device Security Key Marker (MSDR_SDSKM0~3)
  - Primary FFA Key Marker (MSDR_PFFAKM0~3)/Secondary FFA Key Marker (MSDR_SFFAKM0~3)
  - Primary Device Security Control Marker (MSDR_PCTRLM)/Secondary Device Security Control Marker (MSDR_SCTRLM)
  - Primary Configuration Enable Marker (MSDR_PCEM)/ Secondary Configuration Enable Marker (MSDR_SCEM)
- TCFLASH Security Description Records (TCSDRn):
  - TCFLASH Primary Link Key Marker (TCSDRn_PLKM0~3)/TCFLASH Secondary Link Key Marker (TCSDRn_SLKM0~3)
  - TCFLASH Primary Permission Key Marker (TCSDRn_PFPKM0~3)/TCFLASH Secondary Permission Key Marker (TCSDRn_SFPKM0~3)
  - TCFLASH Permission Set Marker (TCSDRn_PFPS0M0~15/TCSDRn_SFPS0M0~15 and
  - TCSDRn_PFPS1M0~15/TCSDRn_SFPS1M0~15)
- EEFLASH Security Description Records (EESDRn):
  - EEFLASH Link Key Marker (EESDR0_LKM0~3)
  - EEFLASH Permission Key Marker (EESDR_FPKM0~3)
  - EEFLASH Permission Set Marker (EESDR_FPS0M0~3 and EESDR_FPS1M0~3)

Memory layout of the BDR

Table 16-1. Memory layout of the BDR

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000008	BDR_SBSM 11111111 11111111 11111111 11111111				BDR_SBMM 11111111 11111111 11111111 11111111			
0x00000010	BDR_DWEM 11111111 11111111 11111111 11111111				reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000018	BDR_ABVEM 11111111 11111111 11111111 11111111				BDR_ABVM 11111111 11111111 11111111 11111111			

Unlike all other record types, the (BDR) is top-address aligned. It is allocated at the end of the Flash sectors.

Interleaving of the sectors is only considered for the aligned block size of the BDR but not for the markers.

Note: BDR is processed in User Mode only and is ignored in Device Test modes.

Memory layout of Main SDR

Table 16-2. Memory layout of the interleaved Main SDR

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	MSDR_SDSEM 11111111 11111111 11111111 11111111				MSDR_PDSEM 11111111 11111111 11111111 11111111			
0x00000008	MSDR_SDSKM0 11111111 11111111 11111111 11111111				MSDR_PDSKM0 11111111 11111111 11111111 11111111			
0x00000010	MSDR_SDSKM1 11111111 11111111 11111111 11111111				MSDR_PDSKM1 11111111 11111111 11111111 11111111			
0x00000018	MSDR_SDSKM2 11111111 11111111 11111111 11111111				MSDR_PDSKM2 11111111 11111111 11111111 11111111			
0x00000020	MSDR_SDSKM3 11111111 11111111 11111111 11111111				MSDR_PDSKM3 11111111 11111111 11111111 11111111			
0x00000028	MSDR_SFFAKM0 11111111 11111111 11111111 11111111				MSDR_PFFAKM0 11111111 11111111 11111111 11111111			
0x00000030	MSDR_SFFAKM1 11111111 11111111 11111111 11111111				MSDR_PFFAKM1 11111111 11111111 11111111 11111111			
0x00000038	MSDR_SFFAKM2 11111111 11111111 11111111 11111111				MSDR_PFFAKM2 11111111 11111111 11111111 11111111			
0x00000040	MSDR_SFFAKM3 11111111 11111111 11111111 11111111				MSDR_PFFAKM3 11111111 11111111 11111111 11111111			
0x00000048	MSDR_SCTRLM 11111111 11111111 11111111 11111111				MSDR_PCTRLM 11111111 11111111 11111111 11111111			
0x00000050	MSDR_SCEM 11111111 11111111 11111111 11111111				MSDR_PCEM 11111111 11111111 11111111 11111111			
0x00000058 - 0x00000078	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							

Even and odd words are interleaved and allocated to different Flash sectors. The even words (sector S0) are allocated to Primary Main Security Configuration. The odd words (sector S1) are allocated to Secondary Main Security Configuration and can be used as shadow configuration when S0 is erased.

Memory Layout of TCFLASH SDR

Table 16-3. Memory layout of TCFLASH SDR

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	TCSDRn_SLKM0 11111111 11111111 11111111 11111111				TCSDRn_PLKM0 11111111 11111111 11111111 11111111			
0x00000008	TCSDRn_SLKM1 11111111 11111111 11111111 11111111				TCSDRn_PLKM1 11111111 11111111 11111111 11111111			
0x00000010	TCSDRn_SLKM2 11111111 11111111 11111111 11111111				TCSDRn_PLKM2 11111111 11111111 11111111 11111111			
0x00000018	TCSDRn_SLKM3 11111111 11111111 11111111 11111111				TCSDRn_PLKM3 11111111 11111111 11111111 11111111			
0x00000020	TCSDRn_SFPKM0 11111111 11111111 11111111 11111111				TCSDRn_PFPKM0 11111111 11111111 11111111 11111111			
0x00000028	TCSDRn_SFPKM1 11111111 11111111 11111111 11111111				TCSDRn_PFPKM1 11111111 11111111 11111111 11111111			
0x00000030	TCSDRn_SFPKM2 11111111 11111111 11111111 11111111				TCSDRn_PFPKM2 11111111 11111111 11111111 11111111			
0x00000038	TCSDRn_SFPKM3 11111111 11111111 11111111 11111111				TCSDRn_PFPKM3 11111111 11111111 11111111 11111111			
0x00000040	TCSDRn_SFPS0M0 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M0 11111111 11111111 11111111 11111111			
0x00000048	TCSDRn_SFPS0M1 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M1 11111111 11111111 11111111 11111111			
0x00000050	TCSDRn_SFPS0M2 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M2 11111111 11111111 11111111 11111111			
0x00000058	TCSDRn_SFPS0M3 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M3 11111111 11111111 11111111 11111111			
0x00000060	TCSDRn_SFPS0M4 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M4 11111111 11111111 11111111 11111111			
0x00000068	TCSDRn_SFPS0M5 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M5 11111111 11111111 11111111 11111111			
0x00000070	TCSDRn_SFPS0M6 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M6 11111111 11111111 11111111 11111111			
0x00000078	TCSDRn_SFPS0M7 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M7 11111111 11111111 11111111 11111111			
0x00000080	TCSDRn_SFPS0M8 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M8 11111111 11111111 11111111 11111111			
0x00000088	TCSDRn_SFPS0M9 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M9 11111111 11111111 11111111 11111111			

Table 16-3. Memory layout of TCFLASH SDR

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000090	TCSDRn_SFPS0M10 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M10 11111111 11111111 11111111 11111111			
0x00000098	TCSDRn_SFPS0M11 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M11 11111111 11111111 11111111 11111111			
0x000000A0	TCSDRn_SFPS0M12 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M12 11111111 11111111 11111111 11111111			
0x000000A8	TCSDRn_SFPS0M13 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M13 11111111 11111111 11111111 11111111			
0x000000B0	TCSDRn_SFPS0M14 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M14 11111111 11111111 11111111 11111111			
0x000000B8	TCSDRn_SFPS0M15 11111111 11111111 11111111 11111111				TCSDRn_PFPS0M15 11111111 11111111 11111111 11111111			
0x000000C0	TCSDRn_SFPS1M0 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M0 11111111 11111111 11111111 11111111			
0x000000C8	TCSDRn_SFPS1M1 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M1 11111111 11111111 11111111 11111111			
0x000000D0	TCSDRn_SFPS1M2 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M2 11111111 11111111 11111111 11111111			
0x000000D8	TCSDRn_SFPS1M3 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M3 11111111 11111111 11111111 11111111			
0x000000E0	TCSDRn_SFPS1M4 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M4 11111111 11111111 11111111 11111111			
0x000000E8	TCSDRn_SFPS1M5 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M5 11111111 11111111 11111111 11111111			
0x000000F0	TCSDRn_SFPS1M6 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M6 11111111 11111111 11111111 11111111			
0x000000F8	TCSDRn_SFPS1M7 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M7 11111111 11111111 11111111 11111111			
0x00000100	TCSDRn_SFPS1M8 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M8 11111111 11111111 11111111 11111111			
0x00000108	TCSDRn_SFPS1M9 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M9 11111111 11111111 11111111 11111111			
0x00000110	TCSDRn_SFPS1M10 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M10 11111111 11111111 11111111 11111111			
0x00000118	TCSDRn_SFPS1M11 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M11 11111111 11111111 11111111 11111111			
0x00000120	TCSDRn_SFPS1M12 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M12 11111111 11111111 11111111 11111111			

Table 16-3. Memory layout of TCFLASH SDR

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000128	TCSDRn_SFPS1M13 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M13 11111111 11111111 11111111 11111111			
0x00000130	TCSDRn_SFPS1M14 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M14 11111111 11111111 11111111 11111111			
0x00000138	TCSDRn_SFPS1M15 11111111 11111111 11111111 11111111				TCSDRn_PFPS1M15 11111111 11111111 11111111 11111111			

Depending on device and Flash macro, even and odd words are interleaved and allocated to different Flash sectors. The even words (sector S0) are allocated to primary markers. The odd words (sector S1) are allocated to secondary markers and can be used as shadow configuration when S0 is erased.

Table 16-4. Availability of Secondary TC SDRs

Device	Flash Macros	Secondary Markers
CY9DF126 Rev 1 CY9DF126 Rev 2 (ATLAS)	TCFLASH0	Supported
	TCFLASH1	Not supported
	EEFLASH0	Not supported
CY9DF125 Rev 1 CY9DF125 Rev 2 CY9DF125 Rev 3 (ATLAS-L)	TCFLASH0	Supported
	TCFLASH1	N.A
	EEFLASH0	Not supported
CY9E226 Rev 1 CY9E226 Rev 2 (TITAN)	TCFLASH0	Supported
	TCFLASH1	N.A.
	EEFLASH0	Not supported

Note: Primary and Secondary markers in TCSDR0 are selected according to the selection of Primary and Secondary MSDR markers (Primary and Secondary device security configuration).

Memory layout of EEFLASH SDRs

Table 16-5. Memory layout of EEFLASH SDR

Offset	+3	+2	+1	+0
0x00000000	EESDR_LKM0 11111111 11111111 11111111 11111111			
0x00000004	EESDR_LKM1 11111111 11111111 11111111 11111111			
0x00000008	EESDR_LKM2 11111111 11111111 11111111 11111111			
0x0000000C	EESDR_LKM3 11111111 11111111 11111111 11111111			
0x00000010	EESDR_FPKM0 11111111 11111111 11111111 11111111			
0x00000014	EESDR_FPKM1 11111111 11111111 11111111 11111111			
0x00000018	EESDR_FPKM2 11111111 11111111 11111111 11111111			
0x0000001C	EESDR_FPKM3 11111111 11111111 11111111 11111111			
0x00000020	EESDR_FPS0M0 11111111 11111111 11111111 11111111			
0x00000024	EESDR_FPS0M1 11111111 11111111 11111111 11111111			
0x00000028	EESDR_FPS0M2 11111111 11111111 11111111 11111111			
0x0000002C	EESDR_FPS0M3 11111111 11111111 11111111 11111111			
0x00000030	EESDR_FPS1M0 11111111 11111111 11111111 11111111			
0x00000034	EESDR_FPS1M1 11111111 11111111 11111111 11111111			
0x00000038	EESDR_FPS1M2 11111111 11111111 11111111 11111111			
0x0000003C	EESDR_FPS1M3 11111111 11111111 11111111 11111111			

Unlike the description records in TCFLASH, the SDR in EEFLASH is not interleaved.

BCR Address Map

Table 16-6. Boot ROM Configuration Address Map for the FCR4 Cluster Series device family

Address	+7	+6	+5	+4	+3	+2	+1	+0	Record	CY9DF126	CY9DF125	CY9EF226
0xB047003C ... 0xB0470000	EEFLASH3:S0								EESDR3	-	-	-
0xB046003C ... 0xB0460000	EEFLASH2:S0								EESDR2	-	-	-
0xB045003C ... 0xB0450000	EEFLASH1:S0								EESDR1	-	-	-
0xB044003C ... 0xB0440000	EEFLASH0:S0 (SA0)								EESDR0	+	+	+
0x00FFFFF8 ... 0x00FFFFF0	TCFLASH0:S7 (SA7)				TCFLASH0:S6 (SA6)				BDR	+	+	+
0x00FF01B8 ... 0x00FF0080	TCFLASH0:S1 (SA1)				TCFLASH0:S0 (SA0)				TCSDR0	+	+	+
0x00FF0078 ... 0x00FF0000									MSDR	+	+	+
0x00FE0138 ... 0x00FE0000	TCFLASH1:S1 (SB1)				TCFLASH1:S0 (SB1)				TCSDR1	+	-	-
0x00FD0138 ... 0x00FD0000	TCFLASH2:S1 (SB1)				TCFLASH2:S0 (SB1)				TCSDR2	-	-	-
0x00FC0138 ... 0x00FC0000	TCFLASH3:S1 (SB1)				TCFLASH3:S0 (SB1)				TCSDR3	-	-	-

Notes:

1- No Flash macro and therefore no description record

2 - Flash macro exists and the description record is supported

The sector names in brackets refer to the (ambiguous) names. The addresses for TCFLASH macros list TCM addresses. The corresponding AXI addresses can be found in the device-specific datasheet. The addresses for EEFLASH macros list the main AXI addresses. For related alternative addresses with different access properties, refer to the device-specific datasheet.

16.2.1 SHE Secure Boot Mode Marker (BDR_SBMM)

Secure Hardware Extension (SHE) command for Secure Boot is issued by the Boot ROM after each reset. The command parameter Secure Boot Size is configured in this marker. The mode parameter selects whether Boot ROM waits for Secure Boot completion or not. The marker is not protected by security settings.

SHE Secure Boot Size Marker (BDR_SBMM)

Figure 16-2. SHE Secure Boot Mode Marker

BDR_SBMM																															
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**
SBMM[31]	SBMM[30]	SBMM[29]	SBMM[28]	SBMM[27]	SBMM[26]	SBMM[25]	SBMM[24]	SBMM[23]	SBMM[22]	SBMM[21]	SBMM[20]	SBMM[19]	SBMM[18]	SBMM[17]	SBMM[16]	SBMM[15]	SBMM[14]	SBMM[13]	SBMM[12]	SBMM[11]	SBMM[10]	SBMM[9]	SBMM[8]	SBMM[7]	SBMM[6]	SBMM[5]	SBMM[4]	SBMM[3]	SBMM[2]	SBMM[1]	SBMM[0]
RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-7. SHE Secure Boot Mode Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SBMM	SHE Secure Boot Mode Marker The value 0x00000000 selects the mode "Measuring during application". The Boot ROM may start the user application before the Secure Boot has finished. A value other than 0x00000000 selects the mode "Measuring before application". The Boot ROM waits for the Secure Boot to finish before starting the user application.

Note: This marker is protected by ECC. Any detected error higher than a 1-bit error will enforce the mode “measuring before application”, forcing the Boot ROM to wait for the secure boot process of SHE if enabled by SHE memory slot BOOT_MAC_KEY.

16.2.2 SHE Secure Boot Size Marker (BDR_SBSM)

Secure Hardware Extension (SHE) command for secure Boot is issued by Boot ROM after each reset. Command parameter Secure Boot Mode is configured in this marker. The size parameter is used to set up the SHE Channel Master. The marker is not protected by securty settings. Note: The Secure Boot Start is set according to BDR_ABVEM and BDR_ABVM. However, it is aligned to the lower 64-bit boundary.

SHE Secure Boot Mode Marker (BDR_SBSM)

Figure 16-3. SHE Secure Boot Size Marker

BDR_SBSM																															
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**
SBSM[31]	SBSM[30]	SBSM[29]	SBSM[28]	SBSM[27]	SBSM[26]	SBSM[25]	SBSM[24]	SBSM[23]	SBSM[22]	SBSM[21]	SBSM[20]	SBSM[19]	SBSM[18]	SBSM[17]	SBSM[16]	SBSM[15]	SBSM[14]	SBSM[13]	SBSM[12]	SBSM[11]	SBSM[10]	SBSM[9]	SBSM[8]	SBSM[7]	SBSM[6]	SBSM[5]	SBSM[4]	SBSM[3]	SBSM[2]	SBSM[1]	SBSM[0]
RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-8. SHE Secure Boot Size Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SBSM	SHE Secure Boot Size Marker This word sets the byte size of the memory area to be verified if the Secure Boot is enabled by the Secure Boot MAC Key. On the one hand, the byte size defines the size for the CMAC algorithm. On the other hand, the 128-bit aligned size is configured to the SHE Channel Master for a data transfer.

Note: This marker is protected by ECC. Any detected error higher than 1-bit error will cancel the SHE secure boot process and enforce secure boot failure if enabled by SHE memory slot BOOT_MAC_KEY. Hence boot protected SHE memory slots are locked against usage.

16.2.3 Debugger Waiting Enable Marker (BDR_DWEM)

The Debugger Waiting Enable Marker delays start of user program after hardware reset or PD2 reset until a debugger is connected or until a maximum waiting time is reached. The marker is not protected by security settings.

Debugger Waiting Enable Marker (BDR_DWEM)

Figure 16-4. Device Security Key Marker

BDR_DWEM																															
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**
DWEM[31]	DWEM[30]	DWEM[29]	DWEM[28]	DWEM[27]	DWEM[26]	DWEM[25]	DWEM[24]	DWEM[23]	DWEM[22]	DWEM[21]	DWEM[20]	DWEM[19]	DWEM[18]	DWEM[17]	DWEM[16]	DWEM[15]	DWEM[14]	DWEM[13]	DWEM[12]	DWEM[11]	DWEM[10]	DWEM[9]	DWEM[8]	DWEM[7]	DWEM[6]	DWEM[5]	DWEM[4]	DWEM[3]	DWEM[2]	DWEM[1]	DWEM[0]
RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-9. Device Security Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DWEM	Debugger Waiting Enable Marker If these bits are set to a value other than 0x292D3A7B, the user program is suspended after a hard/PD2 reset until the debugger is connected or a maximum waiting time is reached. The value 0x292D3A7B immediately starts the user program. Other values the user program waits for the JTAG tool and debugger.

Note:

- The marker is protected by ECC. Any detected error higher than 1-bit error when reading this marker will enforce to wait for JTAG tool and debugger.

There are two phases of waiting for external tool.

1. During the first waiting period, presence of JTAG tool hardware is tested. If no JTAG tool is found after 19200 RC-Clock cycles (+5%), user programe is started. In this case, delay is about 0.8 ms to 3.3 ms at 24 MHz to 6 MHz RC-Clock.
2. If JTAG tool was found, a second waiting period for reception of acknowledge from host application (debugger) starts. If not recieved, waiting is cancelled after 2^23 RC-Clock cycles. In this case delay is about 350 ms to 1.4 sec at 24 MHz to 6 MHz RC-clock.

The acknowledgment informs MCU that host application has transfered all necessary data (e.g. Security keys, breakpoint configuration) and is ready to execute user program.

If tooling is used, the actual delay untill connection, depends on external JTAG clock frequency and amount of data to be transfered,

In case of software reset, all tool related configuration data are preserved. User program is started immediately without waiting for JTAG tool.

16.2.4 Alternative Boot Vector Marker (BDR_ABVM)

The Alternative Boot vector Enable Marker stores the secondary start address of user program. It is used if BDR_ABVEM is set. The marker is not protected by security settings.

Alternative Boot Vector Marker (BDR_ABVM)

Figure 16-5. Device Security Key Marker

BDR_ABVM																															
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**
ABVM[31]	ABVM[30]	ABVM[29]	ABVM[28]	ABVM[27]	ABVM[26]	ABVM[25]	ABVM[24]	ABVM[23]	ABVM[22]	ABVM[21]	ABVM[20]	ABVM[19]	ABVM[18]	ABVM[17]	ABVM[16]	ABVM[15]	ABVM[14]	ABVM[13]	ABVM[12]	ABVM[11]	ABVM[10]	ABVM[9]	ABVM[8]	ABVM[7]	ABVM[6]	ABVM[5]	ABVM[4]	ABVM[3]	ABVM[2]	ABVM[1]	ABVM[0]
RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-10. Device Security Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ABVM	Alternative Boot Vector Marker These bits store the alternative user program start address.

Note:

- This marker is protected by ECC. Any detected error higher than 1-bit error will deactivate the alternative address. If the alternative address is configured (e.g to start a boot loader) a fall-back code at the default address should implement defined behaviour.
- User program is always started in Arm state.

16.2.5 Alternative Boot Vector Enable Marker (BDR_ABVEM)

The Alternative Boot Vector Enable Marker is a Magic Word marker, which activates the secondary user program start address as stored in BDR_ABVM. The marker is not protected by security settings.

Alternative Boot Vector Enable Marker (BDR_ABVEM)

Figure 16-6. Device Security Key Marker

BDR_ABVEM																															
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**
ABVEM[31]	ABVEM[30]	ABVEM[29]	ABVEM[28]	ABVEM[27]	ABVEM[26]	ABVEM[25]	ABVEM[24]	ABVEM[23]	ABVEM[22]	ABVEM[21]	ABVEM[20]	ABVEM[19]	ABVEM[18]	ABVEM[17]	ABVEM[16]	ABVEM[15]	ABVEM[14]	ABVEM[13]	ABVEM[12]	ABVEM[11]	ABVEM[10]	ABVEM[9]	ABVEM[8]	ABVEM[7]	ABVEM[6]	ABVEM[5]	ABVEM[4]	ABVEM[3]	ABVEM[2]	ABVEM[1]	ABVEM[0]
RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf	RfWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-11. Device Security Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ABVEM	Alternative Boot Vector Enable Marker If these bits are set to a Magic Word value of 0x292D3A7B, the user program is started at the address stored in BDR_ABVM. Other values start the user program at the default address 0x00800000.

Note:

- This marker is protected by ECC. Any detected error higher than 1-bit error will deactivate the alternative address. If the alternative address is configured (e.g. to start a boot loader), a fall-back code of application at the default address should implement defined behaviour.

16.2.6 Primary Device Security Enable Marker (MSDR_PDSEM)/ Secondary Device Security Enable Marker (MSDR_SDSEM)

The primary Device Security Enable Marker MSDR_PDSEM is the main switch for the device security. It controls whether access via JTAG is restricted or whether Flash contents can be read. The marker cannot be read back from Flash after Boot ROM has passed. MSDR_PDSEM is active only if Primary Main security Configuration (Sector S0) is selected by PCEM. Otherwise if SCEM selects Secondary Main security configuration. Secondary Device Security enable Marker SDSEM replaces PDSEM.

Primary Device Security Enable Marker (MSDR_PDSEM) and Secondary Device Security Enable Marker (MSDR_SDSEM)

Figure 16-7. Primary Device Security Enable Marker

MSDR_PDSEM																															
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**
PDSEM[31]	PDSEM[30]	PDSEM[29]	PDSEM[28]	PDSEM[27]	PDSEM[26]	PDSEM[25]	PDSEM[24]	PDSEM[23]	PDSEM[22]	PDSEM[21]	PDSEM[20]	PDSEM[19]	PDSEM[18]	PDSEM[17]	PDSEM[16]	PDSEM[15]	PDSEM[14]	PDSEM[13]	PDSEM[12]	PDSEM[11]	PDSEM[10]	PDSEM[9]	PDSEM[8]	PDSEM[7]	PDSEM[6]	PDSEM[5]	PDSEM[4]	PDSEM[3]	PDSEM[2]	PDSEM[1]	PDSEM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-12. Primary Device Security Enable Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PDSEM	Primary Device Security Enable Marker If any bit has a value of "0", device security is activated and the security bridge is configured. '0xFFFFFFFF':Security is deactivated if all bits belonging to MSDR_PDSKM are '1'. '0x99999999': Security is activated. Other values: Not recommended even if it activates security. PDSEM is selected by PCEM.

Figure 16-8. Secondary Device Security Enable Marker

MSDR_SDSEM																															
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**
SDSEM[31]	SDSEM[30]	SDSEM[29]	SDSEM[28]	SDSEM[27]	SDSEM[26]	SDSEM[25]	SDSEM[24]	SDSEM[23]	SDSEM[22]	SDSEM[21]	SDSEM[20]	SDSEM[19]	SDSEM[18]	SDSEM[17]	SDSEM[16]	SDSEM[15]	SDSEM[14]	SDSEM[13]	SDSEM[12]	SDSEM[11]	SDSEM[10]	SDSEM[9]	SDSEM[8]	SDSEM[7]	SDSEM[6]	SDSEM[5]	SDSEM[4]	SDSEM[3]	SDSEM[2]	SDSEM[1]	SDSEM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-13. Secondary Device Security Enable Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SDSEM	Secondary Device Security Enable Marker If any bit has a value of "0", device security is activated and the security bridge is configured. '0xFFFFFFFF':Security is deactivated if all bits belonging to MSDR_SDSKM are '1'. '0x99999999': Security is activated. Other values: Not recommended even if it activates security. SDSEM is selected by SCEM.

Note: This marker is protected by ECC. Any detected error in PDSEM or SDSEM higher than a 1-bit error will invalidate all primary and secondary markers respectively.

16.2.7 Primary Device Security Key Marker (MSDR_PDSKM0~3)/ Secondary Device Security Key Marker (MSDR_SDSKM0~3)

The Primary Device Security key PDSKM0~3 and Secondary Device security Key SDSKM0~3 consist of 128 bits each and are formed by four separated marker words. It cannot be read back from Flash. PDSKM0~3 is active only if selected by PCEM but if SCEM selects the Secondary Main Security Configuration, than the Secondary Device Security Enable Marker SDSKM0~3 replaces PDSKM0~3. Device Security Key Markers hold the nominal value of the key for unlocking Device security if allowed by MSDR_PCTRLM:JTAGEN or MSDR_PCTRLM:JTAGSWEN. With this key, access is granted to all resources including internal Flash.

Primary Device security Key Marker (MSDR_PDSKM0~3) and Secondary Device Security Key Marker (MSDR_SDSKM0~3)

Figure 16-9. Primary Device Key Marker

MSDR_PDSKM0																															
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**
PDSKM[31]	PDSKM[30]	PDSKM[29]	PDSKM[28]	PDSKM[27]	PDSKM[26]	PDSKM[25]	PDSKM[24]	PDSKM[23]	PDSKM[22]	PDSKM[21]	PDSKM[20]	PDSKM[19]	PDSKM[18]	PDSKM[17]	PDSKM[16]	PDSKM[15]	PDSKM[14]	PDSKM[13]	PDSKM[12]	PDSKM[11]	PDSKM[10]	PDSKM[9]	PDSKM[8]	PDSKM[7]	PDSKM[6]	PDSKM[5]	PDSKM[4]	PDSKM[3]	PDSKM[2]	PDSKM[1]	PDSKM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-14. Primary Device Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PDSKM	Primary Device Security Key Marker This marker holds the key for unlocking device security if allowed by the marker bits MSDR_PDCTRLM:JTAGEN or MSDR_PDCTRLM:JTAGSWEN. The key may have any value. However, for unsecured devices all bits must be '1'. All bits 1: This is a valid but trivial key if PDSEM enables security. It deactivates security if all PDSEM bits are '1'. Other values: Security is activated even if all bits of PDSEM are '1'. The unsecured default value, the secure default value and the secure value for this register value is "Unused." PDSKM is selected by PCEM.

Figure 16-10. Secondary Device Key Marker

MSDR_SDSKM0																															
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**
SDSKM[31]	SDSKM[30]	SDSKM[29]	SDSKM[28]	SDSKM[27]	SDSKM[26]	SDSKM[25]	SDSKM[24]	SDSKM[23]	SDSKM[22]	SDSKM[21]	SDSKM[20]	SDSKM[19]	SDSKM[18]	SDSKM[17]	SDSKM[16]	SDSKM[15]	SDSKM[14]	SDSKM[13]	SDSKM[12]	SDSKM[11]	SDSKM[10]	SDSKM[9]	SDSKM[8]	SDSKM[7]	SDSKM[6]	SDSKM[5]	SDSKM[4]	SDSKM[3]	SDSKM[2]	SDSKM[1]	SDSKM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-15. Secondary Device Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SDSKM	Secondary Device Security Key Marker This marker holds the key for unlocking device security if allowed by the marker bits MSDR_SDCTRLM:JTAGEN or MSDR_SDCTRLM:JTAGSWEN. The key may have any value. However, for unsecured devices all bits must be '1'. All bits 1: This is a valid but trivial key if SDSEM enables security. It deactivates security if all SDSEM bits are '1'. Other values: Security is activated even if all bits of SDSEM are '1'. The unsecured default value, the secure default value and the secure value for this register value is "Unused." SDSKM is selected by SCEM.

Notes:

- These markers are stored in separated words with gaps of one word each. This ensures storage in same Flash sector even if two sectors are interleaved to double words.In case security is acivated by MSDR_PDSEM = 0x99999999 respectively. MSDR_SDSEM = 0x 99999999, the key value for all bytes 0xFF is trivial (0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF). the key value for all bytes 0xFF is trivial However it might be used for testing the flow of updating Flash by user software or by external tool.
- The trivial PDSKM/SDSKM key (all bytes FF) is not written to the Security Checker Register if security mode is evaluated to OFF.
- Values other than the trivial code set security mode ON even if MSDR_PDSEM/MSDR_SDSEM is set to 0xFFFFFFFF.

Table 16-16. Device Security Marker Codes

Marker Value		
MSDR_PDSEM/ MSDR_SDSEM	**MSDR_PDSKM/MSDR_SDSKM**	**Security Mode**
0xFFFFFFFF	0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	OFF
0x99999999	-	ON
Other combinations (not recommended)		(ON)

16.2.8 Primary FFA Key Marker (MSDR_PFFAKM0~3)/Secondary FFA Key Marker (MSDR_SFFAKM0~3)

The Primary Field Failure Analysis Key Marker (PFFAKM0~3) consists of 128 bits and is formed by four separated marker words. It cannot be read back from Flash. This marker is only active if Primary Main Security Configuration (Sector S0) is selected by PCEM. Otherwise, if SCEM selects Secondary Main Security Configuration, Secondary Device Security Enable Marker (SFFAKM0~3) replaces PFFAKM0~3. The Field Failure Analysis Key is an alternative to the Device Securtiy key. It holds the key value for unlocking Device Security if allowed by MSDR_PCTRLM:JTAGEN or MSDR_PCTRLM:JTAGSWEN. Unlike the Device Security Key, It does not grant full access to all resources but hides internal Flash memory from reading.

Primary FFA Key Marker (MSDR_PFFAKM0~3) and Secondary FFA Key Marker (MSDR_SFFAKM0~3)

Figure 16-11. Primary FFA Key Marker

MSDR_PFFAKM0																															
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**
PFFAKM0[31]	PFFAKM0[30]	PFFAKM0[29]	PFFAKM0[28]	PFFAKM0[27]	PFFAKM0[26]	PFFAKM0[25]	PFFAKM0[24]	PFFAKM0[23]	PFFAKM0[22]	PFFAKM0[21]	PFFAKM0[20]	PFFAKM0[19]	PFFAKM0[18]	PFFAKM0[17]	PFFAKM0[16]	PFFAKM0[15]	PFFAKM0[14]	PFFAKM0[13]	PFFAKM0[12]	PFFAKM0[11]	PFFAKM0[10]	PFFAKM0[9]	PFFAKM0[8]	PFFAKM0[7]	PFFAKM0[6]	PFFAKM0[5]	PFFAKM0[4]	PFFAKM0[3]	PFFAKM0[2]	PFFAKM0[1]	PFFAKM0[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-17. Primary FFA Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PFFAKM0	Primary FFA Key Marker This marker holds the key for partially unlocking device security if allowed by marker bits MSDR_PCTRLM:JTAGEN or MSDR_PCTRLM:JTAGSWEN. The key may have any value: DSKM: If set to the same value as MSDR_PDSKM, limited access cannot be granted. Other values: An alternative key grants access to all other resources other than Flash. PDSEM is selected by PCEM.

Figure 16-12. Secondary FFA Key Marker

MSDR_SFFAKM0																															
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**
SFFAKM[31]	SFFAKM[30]	SFFAKM[29]	SFFAKM[28]	SFFAKM[27]	SFFAKM[26]	SFFAKM[25]	SFFAKM[24]	SFFAKM[23]	SFFAKM[22]	SFFAKM[21]	SFFAKM[20]	SFFAKM[19]	SFFAKM[18]	SFFAKM[17]	SFFAKM[16]	SFFAKM[15]	SFFAKM[14]	SFFAKM[13]	SFFAKM[12]	SFFAKM[11]	SFFAKM[10]	SFFAKM[9]	SFFAKM[8]	SFFAKM[7]	SFFAKM[6]	SFFAKM[5]	SFFAKM[4]	SFFAKM[3]	SFFAKM[2]	SFFAKM[1]	SFFAKM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-18. Secondary FFA Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SFFAKM	Secondary FFA Key Marker This marker holds the key for partially unlocking device security if allowed by marker bits MSDR_SCTRLM:JTAGEN or MSDR_SCTRLM:JTAGSWEN. The key may have any value: DSKM: If set to the same value as MSDR_SDSKM, limited access cannot be granted. Other values: An alternative key grants access to all other resources other than Flash. SDSEM is selected by SCEM.

Notes:

- This marker is stored as separate words with gaps of one word each. This ensures storage in the same Flash sector even if two sectors are interleaved to double words.
- The value 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF is trivial but valid. It might be used to protect the customer IP in Flash memory. If there is no need to protect access to other resources (RAM, IO, CPU), it is recommended.
- Unlike MSDR_PDSKM/MSDR_SDSKM, any other MSDR_PFFAKM/MSDR_SFFAKM value other than the trivial one does not activate security if it has not already been set by MSDR_PDSEM/MSDR_SDSEM.
- The key is not written to the Security Checker register if Security mode is evaluated to OFF.

16.2.9 Primary Device Security Control Marker (MSDR_PCTRLM)/Secondary Device Security Control Marker (MSDR_SCTRLM)

The Primary Device Secuity Control Marker (PCTRLM) adjusts the security feature in detail. It consists of single bits per function. It cannot be read back from Flash. This marker is only active if Primary Main Security Configuration (Sector S0) is selected by PCEM. Otherwise, if SCEM selects Secondary Main Security Configuration, Secondary Device Security Control Marker (SCTRLM) replaces PCTRLM.

Primary Device Security Control Marker (MSDR_PCTRLM) and Secondary Device Security Control Marker (MSDR_SCTRLM)

Figure 16-13. Primary Device Security Control Marker

MSDR_PCTRLM																														
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**
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EEPEN	TCPEN	-	EEMEEN	TCMEEN	DFPPEN	-	JTAGSWEN	JTAGEN	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RfdWf	RfdWf	-	RfdWf	RfdWf	RfdWf	-	RfdWf	RfdWf	-
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-19. Primary Device Security Control Marker bits

Bit Position	Bit Field Name	Bit Description
[31:16]	-	-
[15:10]	-	-
[9]	EEPEN	EEFLASH Permission Enable In User Mode, EEFLASH Permission Enable loads two alternative Flash-macro-wise sets of sector-wise access permission flags and activates the first set. '0': Do not load and activate permission sets. This grants full access to application. '1': Load and activate permission sets. EEPEN is ignored in device test modes.

Table 16-19. Primary Device Security Control Marker bits

Bit Position	Bit Field Name	Bit Description
[8]	TCPEN	TCFLASH Permission Enable In User Mode, TCFLASH Permission Enable loads two alternative Flash-macro-wise sets of sector-wise access permission flags and activates the first set. '0': Do not load and activate permission sets. This grants full access to application. '1': Load and activate permission sets. TCPEN is ignored in device test modes.
[7]	-	-
[6]	EEMEEN	EEFLASH Macro Erase Enable This bit allows entire EEFLASH macros of the secured device to be erased in test modes. '0': EEFLASH macros cannot be erased. '1': EEFLASH macros can be erased. EEMEEN is effective in test mode only and if the device is secured. All current devices ignore EEMEEN in Flash Parallel Programming mode and enable erasing. In User mode, access protection is defined by permission sets.
[5]	TCMEEN	TCFLASH Macro Erase Enable This bit allows the entire TCFLASH macros of the secured device to be erased in test modes. '0': TCFLASH macros cannot be erased. '1': TCFLASH macros can be erased. TCMEEN is effective in Test Modes only and if device is secured. All current devices ignore TCMEEN in Flash Parallel Programming mode and enable erasing. In User mode, access protection is defined by permission sets.
[4]	DFPPEN	Direct FPP Enable This bit allows Flash Parallel Programming Mode to be entered without transferring a key to the secured device in Board Test Mode. '0': FPP cannot be entered without the key. '1': FPP can be entered without key. All current devices ignore DFPPEN and always allow FPP. This allows the recovery of wrongly configured devices by erasing Flash. Secured contents, however, cannot be read.
[3]	-	-

Table 16-19. Primary Device Security Control Marker bits

Bit Position	Bit Field Name	Bit Description
[2]	JTAGSWEN	JTAG Software Enable This bit allows the user program to grant JTAG key transfer using MSDR_SCCTRL:JTAGSWEN if the device is secured. '0': Software cannot change MSDR_SCCTRL:JTAGSWEN. Access rights as set by the marker bit MSDR_JTAGEN are fixed. '1': Software can write SCCTRL:JTAGSWEN once and grant access rights set by setting SCCTRL:JTAGSWEN = '1'. If the device is unsecured, the marker bit JTAGSWEN is ignored.
[1]	JTAGEN	JTAG Enable This bit grants permission to an external tool to transfer the key to the secured device. '0': JTAG key transfer is disabled. The device cannot be unlocked. '1': JTAG key transfer is enabled. If MSDR_PCTRLM:JTAGEN is set to '0' and MSDR_PCTRLM:JTAGSWEN is set '1', the user program can still grant key transfer when MSDR_SCCTRL:JTAGSWEN = '1'. If the device is unsecured, MSDR_PCTRLM:JTAGEN is ignored.
[0]	-	-

Note: Successfully erased SDR results in enabled Flash permission in marker and will force to grant full access by application to all sectors in User Mode.

Figure 16-14. Secondary Device Security Control Marker

MSDR_SCTRLM																															
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
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EEPEN	TCPEN	-	EEMEEN	TCMEEN	DFPPEN	-	JTAGSWEN	JTAGEN	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RfdWf	RfdWf	-	RfdWf	RfdWf	RfdWf	-	RfdWf	RfdWf	-
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Note: 1* means that permission is enabled but permission sets are configured to restrictive values instead of user configuration.

Table 16-20. Secondary Device Security Control bits

Bit Position	Bit Field Name	Bit Description
[31:16]	-	-
[15:10]	-	-
[9]	EEPEN	EEFLASH Permission Enable In User Mode, EEFLASH Permission Enable loads two alternative Flash-macro-wise sets of sector-wise access permission flags and activates the first set. '0': Do not load and activate permission sets. This grants full access to application. '1': Load and activate permission sets. EEPEN is ignored in device test modes.
[8]	TCPEN	TCFLASH Permission Enable In User Mode, TCFLASH Permission Enable loads two alternative Flash-macro-wise sets of sector-wise access permission flags and activates the first set. '0': Do not load and activate permission sets. This grants full access to application. '1': Load and activate permission sets. TCPEN is ignored in device test modes.
[7]	-	-
[6]	EEMEEN	EEFLASH Macro Erase Enable This bit allows entire EEFLASH macros of the secured device to be erased in test modes. '0': EEFLASH macros cannot be erased. '1': EEFLASH macros can be erased. EEMEEN is effective in test mode only and if the device is secured. All current devices ignore EEMEEN in Flash Parallel Programming mode and enable erasing. In User mode, access protection is defined by permission sets.
[5]	TCMEEN	TCFLASH Macro Erase Enable This bit allows the entire TCFLASH macros of the secured device to be erased in test modes. '0': TCFLASH macros cannot be erased. '1': TCFLASH macros can be erased. TCMEEN is effective in Test Modes only and if device is secured. All current devices ignore TCMEEN in Flash Parallel Programming mode and enable erasing. In User mode, access protection is defined by permission sets.

Table 16-20. Secondary Device Security Control bits

Bit Position	Bit Field Name	Bit Description
[4]	DFPPEN	Direct FPP Enable This bit allows Flash Parallel Programming Mode to be entered without transferring a key to the secured device in Board Test Mode. '0': FPP cannot be entered without the key. '1': FPP can be entered without key. All current devices ignore DFPPEN and always allow FPP. This allows the recovery of wrongly configured devices by erasing Flash. Secured contents, however, cannot be read.
[3]	-	-
[2]	JTAGSWEN	JTAG Software Enable This bit allows the user program to grant JTAG key transfer using MSDR_SCCTRL:JTAGSWEN if the device is secured. '0': Software cannot change MSDR_SCCTRL:JTAGSWEN. Access rights as set by the marker bit MSDR_JTAGEN are fixed. '1': Software can write SCCTRL:JTAGSWEN once and grant access rights set by setting SCCTRL:JTAGSWEN = '1'. If the device is unsecured, the marker bit JTAGSWEN is ignored.
[1]	JTAGEN	JTAG Enable This bit grants permission to an external tool to transfer the key to the secured device. '0': JTAG key transfer is disabled. The device cannot be unlocked. '1': JTAG key transfer is enabled. If MSDR_PCTRLM:JTAGEN is set to '0' and MSDR_PCTRLM:JTAGSWEN is set '1', the user program can still grant key transfer when MSDR_SCCTRL:JTAGSWEN = '1'. If the device is unsecured, MSDR_PCTRLM:JTAGEN is ignored.
[0]	-	-

Note: Successfully erased SDR results in enabled Flash permission in marker and will force to grant full access by application to all sectors in User Mode.

16.2.10 Primary Configuration Enable Marker (MSDR_PCEM)/ Secondary Configuration Enable Marker (MSDR_SCEM)

The Primary configuration enable Marker (PCEM) and Secondary Configuration Marker (SCEM) perserve security mode ON even if a sector with MSDR markers is erased by running the user program. While one sector is erased, the security configuration is stored in an alternative sector.

The markers PCEM and SCEM allow user program to prepare an alterative sector, to verify its validity and to activate it. The Primary configuration Enable Marker selects the Primary Main Security configuration in Sector S0. If Primary Main Security Configuration is not selected or invalid, the Boot ROM will process primary configuration data. The configuration is valid if no ECC errors occur while reading it. If primary Main Security Configuration is selected and valid, Boot ROM will check the selection and validity of Secondary Main Security Configuration in sector S1. If the Secondary Main Security Configuration is selected by MSDR_SCEM and valid, the Boot ROM will process secondary configuration data. If neither Primary nor Secondary configuration is selected but all Primary markers are equal 0xFFFFFFFF and valid, MSDR is considered empty and security mode is OFF.

If neither Primary nor Secondary configuration is selected and valid and if the Primary configuration is not empty, all security configurations are considered corrupted, security state is ON and security configuration is set to strict values. In User Mode, user program is not authorised and the device is reset.

Primary Configuration Enable Marker (MSDR_PCEM) and Secondary Configuration Enable Marker (MSDR_SCEM)

Figure 16-15. Primary Configuration Enable Marker

MSDR_PCEM																															
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**
PCEM[31]	PCEM[30]	PCEM[29]	PCEM[28]	PCEM[27]	PCEM[26]	PCEM[25]	PCEM[24]	PCEM[23]	PCEM[22]	PCEM[21]	PCEM[20]	PCEM[19]	PCEM[18]	PCEM[17]	PCEM[16]	PCEM[15]	PCEM[14]	PCEM[13]	PCEM[12]	PCEM[11]	PCEM[10]	PCEM[9]	PCEM[8]	PCEM[7]	PCEM[6]	PCEM[5]	PCEM[4]	PCEM[3]	PCEM[2]	PCEM[1]	PCEM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-21. Primary Configuration Enable Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PCEM	Configuration Enable Marker The Configuration Enable Marker activates the Main Security Configuration in the same Flash sector. 0x292D3A7B: Main Security Configuration is selected. Other values: Main Security Configuration is not selected. If PCEM and SCEM are set to '0x292D3A7B', then the primary configuration is selected.

Figure 16-16. Secondary Configuration Enable Marker

MSDR_SCEM																															
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**
SCEM[31]	SCEM[30]	SCEM[29]	SCEM[28]	SCEM[27]	SCEM[26]	SCEM[25]	SCEM[24]	SCEM[23]	SCEM[22]	SCEM[21]	SCEM[20]	SCEM[19]	SCEM[18]	SCEM[17]	SCEM[16]	SCEM[15]	SCEM[14]	SCEM[13]	SCEM[12]	SCEM[11]	SCEM[10]	SCEM[9]	SCEM[8]	SCEM[7]	SCEM[6]	SCEM[5]	SCEM[4]	SCEM[3]	SCEM[2]	SCEM[1]	SCEM[0]
RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf	RfdWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-22. Secondary Configuration Enable Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SCEM	Configuration Enable Marker The Configuration Enable Marker activates the Main Security Configuration in the same Flash sector. 0x292D3A7B: Main Security Configuration is selected. Other values: Main Security Configuration is not selected. If PCEM and SCEM are set to '0x292D3A7B', then the primary configuration is selected.

Note: Wrong sequence in using PCEM and SCEM by user program, boot loader or JTAG programmer may cause the device getting inaccesible after sudden reset. In such case only erasing whole device in parallel Programming Mode can recover it.

16.2.11 TCFLASH Primary Link Key Marker (TCSDRn_PLKM0~3)/ TCFLASH Secondary Link Key Marker (TCSDRn_SLKM0~3)

The link key consist of 128 bits and is formed by four markers words. It is stored to each Flash macro. In case of interleaved TCFLASH sectors, the words are separated by gaps, It is not readable in Flash. It is processed independently of actual security setting in MSDR.

This marker holds the key for securing Flash macros for the case of erasing them. Unlike the security keys, this key is not configured for later transfer by external tool or by user program. The Link Keys are compared before user program is started. The result of comparison can be read from Security Checker register SCSTAT0.

For this marker, ‘n’ refers to instances 1 and 2.

TCFLASH Link Key Marker (TCSDRn_PLKM0~3/TCSDRn_SLKM0~3)

Figure 16-17. Primary TCFLASH Link Key Marker

TCSDRn_PLKM0																															
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**
LKM[31]	LKM[30]	LKM[29]	LKM[28]	LKM[27]	LKM[26]	LKM[25]	LKM[24]	LKM[23]	LKM[22]	LKM[21]	LKM[20]	LKM[19]	LKM[18]	LKM[17]	LKM[16]	LKM[15]	LKM[14]	LKM[13]	LKM[12]	LKM[11]	LKM[10]	LKM[9]	LKM[8]	LKM[7]	LKM[6]	LKM[5]	LKM[4]	LKM[3]	LKM[2]	LKM[1]	LKM[0]
RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-23. Primary TCFLASH Link Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	LKM	Link Key Marker This marker holds the key for connecting the security of multiple Flash macros. = TCSDR0_LKM: If another macro other than TCFLASH0 has the same LKM value as TCFLASH0, then its code can be executed. != TCSDR0_LKM: If another macro other than TCFLASH0 has a different LKM value to TCFLASH0, its code cannot be executed. There is no special key value with a dedicated meaning.

Figure 16-18. Secondary TCFLASH Link Key Marker

TCSDRn_SLKM0																															
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**
LKM[31]	LKM[30]	LKM[29]	LKM[28]	LKM[27]	LKM[26]	LKM[25]	LKM[24]	LKM[23]	LKM[22]	LKM[21]	LKM[20]	LKM[19]	LKM[18]	LKM[17]	LKM[16]	LKM[15]	LKM[14]	LKM[13]	LKM[12]	LKM[11]	LKM[10]	LKM[9]	LKM[8]	LKM[7]	LKM[6]	LKM[5]	LKM[4]	LKM[3]	LKM[2]	LKM[1]	LKM[0]
RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-24. Secondary TCFLASH Link Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	LKM	Link Key Marker This marker holds the key for connecting the security of multiple Flash macros. = TCSDR0_LKM: If another macro other than TCFLASH0 has the same LKM value as TCFLASH0, then its code can be executed. != TCSDR0_LKM: If another macro other than TCFLASH0 has a different LKM value to TCFLASH0, its code cannot be executed. There is no special key value with a dedicated meaning.

Note: This maker is stored in separated words. In the case of TCFLASH, there are gaps of one word between each marker word. This ensures storage in the same Flash sector even if two sectors are interleaved to double words.

16.2.12 TCFLASH Primary Permission Key Marker (TCSDRn_PFPKM0~3)/TCFLASH Secondary Permission Key Marker (TCSDRn_SFPKM0~3)

The Flash Permission Key Marker is formed by four markers words. It consist of 127 bis key value. The marker is stored to each Flash macro. In case of TCFLASH macros, the words are separated. It is only used in User Mode but is not readable in all device modes. This marker holds the key, which allows user software or debug tool to switch Primary and Secondary Flash Permission sets in a secure but also safe way. Only if the correct key is known to the code, it can select alternative Permission set and program otherwise write protected sectors. To use the key, refer to the description of the Flash Permission User Key Registers SCTCPUSRKEY and SCEEPUSRKEY of Security Checker. Note that Flash Permission Key Marker does not implement corresponding bit 128 of SCTCPUSRKEY and SCEEPUSRKE Y. When loaded by Boot ROM, the permission set is selected according to hardware default.

Flash Permission Key Marker is processed independently of actual security setting in MSDR_PDSEM/MSDR/SDSEM but it can be controlled by the marker MSDR_PCTRLM/ MSDR_SCTRLM.

- All flash Permission Key Markers stored in a TCFLASH macro can be enabled or disabled by MSDR_CTRLM:TCPEN.
- All Flash Permission Key Markers stored in a EEFLASH macro can be enabled or disabled by MSDR_CTRLM:EEPEN.

If Flash Permissions are enabled by TCPREN and EEPEN respectively, the Boot ROM will copy the flash Permission Key and Flash Permission sets to the Security Checker and activate the configured set. If it is disabled, the Boot ROM will not configure the Permission Key and Flash Permission sets, instead leaving that to the user software.

TCFlash Permission Key Marker (TCSDRn_PFPKM0~3/TCSDRn_SFPKM0~3)

Figure 16-19. Primary TCFlash Permission Key

TCSDRn_PFPKM0																															
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**
FPKM[31]	FPKM[30]	FPKM[29]	FPKM[28]	FPKM[27]	FPKM[26]	FPKM[25]	FPKM[24]	FPKM[23]	FPKM[22]	FPKM[21]	FPKM[20]	FPKM[19]	FPKM[18]	FPKM[17]	FPKM[16]	FPKM[15]	FPKM[14]	FPKM[13]	FPKM[12]	FPKM[11]	FPKM[10]	FPKM[9]	FPKM[8]	FPKM[7]	FPKM[6]	FPKM[5]	FPKM[4]	FPKM[3]	FPKM[2]	FPKM[1]	FPKM[0]
RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-25. Primary TCFlash Permission Key bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FPKM	Flash Permission Key Marker These bits hold the 127 bits of the key. They correspond to the bits of registers SCTCPUSRKEY and SCEEPUSRKEY. These bits hold the secret key for switching permission, which is set and known only to the legal user software. There is no special key value with a dedicated meaning.

Figure 16-20. Secondary TCFlash Permission Key

TCSDRn_SFPKM0																															
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**
FPKM[31]	FPKM[30]	FPKM[29]	FPKM[28]	FPKM[27]	FPKM[26]	FPKM[25]	FPKM[24]	FPKM[23]	FPKM[22]	FPKM[21]	FPKM[20]	FPKM[19]	FPKM[18]	FPKM[17]	FPKM[16]	FPKM[15]	FPKM[14]	FPKM[13]	FPKM[12]	FPKM[11]	FPKM[10]	FPKM[9]	FPKM[8]	FPKM[7]	FPKM[6]	FPKM[5]	FPKM[4]	FPKM[3]	FPKM[2]	FPKM[1]	FPKM[0]
RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-26. Secondary TCFlash Permission Key bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FPKM	Flash Permission Key Marker These bits hold the 127 bits of the key. They correspond to the bits of registers SCTCPUSRKEY and SCEEPUSRKEY. These bits hold the secret key for switching permission, which is set and known only to the legal user software. There is no special key value with a dedicated meaning.

Notes:

- On CY9DF126 (“Atlas”) the Flash Permission Key marker TCSDR1_FPKM is ignored. Both interleaved Flash macros are controlled by the same register SCTCUSRKEY.

16.2.13 TCFLASH Permission Set Marker (TCSDRn_PFPS0M0~15/ TCSDRn_SFPS0M0~15 and TCSDRn_PFPS1M0~15/TCSDRn_SFPS1M0~15)

A TCFLASH Permission Set Marker is formed by sixteen marker words. It consists of 128 groups of three bits. Two of these markers are stored in each TCFLASH macro.

This marker holds a collection of sector-wise flags to enable reading, execution and writing (erasing) of the corresponding sector. Two of these sets are provided for each macro, which can change the permission profile during run time. The alternative set can be selected by bit 127 of register SCCFG_SCTCPUSRKEY.

Flash permission Set Markers are processed independently of actual security setting in MSDR_PDSEM/MSDR_SDSEM but can be controlled by the marker MSDR_PCTRLM/ MSDR_SCTRLM. All Flash Permission Set Marker stored in a TCFLASH macro can be enabled or disabled by marker bit TCPEN in PCTRLM/SCTRLM.

If Flash Permissions are enabled by TCPEN, the Boot ROM will copy the Flash Permission Key and the Flash Permission Key Marker.

If it is disabled, the Boot ROM will not configure Permission Keys and Flash Permission sets, instead leaving it to the user software.

TCFLASH Permission Set Marker (TCSDRn_PFPS0M0~15/TCSDRn_SFPS0M0~15 and TCSDRn_PFPS1M0~15/TCSDRn_SFPS1M0~15)

Figure 16-21. Primary Flash Permission Set Markers

TCSDRn_PFPS0M0																															
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**
-	READ7	WRITE7	EXEC7	-	READ6	WRITE6	EXEC6	-	READ5	WRITE5	EXEC5	-	READ4	WRITE4	EXEC4	-	READ3	WRITE3	EXEC3	-	READ2	WRITE2	EXEC2	-	READ1	WRITE1	EXEC1	-	READ0	WRITE0	EXEC0
-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[31]	-	-
[30]	READ7	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[29]	WRITE7	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[28]	EXEC7	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[27]	-	-
[26]	READ6	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[25]	WRITE6	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[24]	EXEC6	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[23]	-	-
[22]	READ5	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[21]	WRITE5	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[20]	EXEC5	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[19]	-	-

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[18]	READ4	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[17]	WRITE4	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[16]	EXEC4	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[15]	-	-
[14]	READ3	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[13]	WRITE3	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[12]	EXEC3	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[11]	-	-
[10]	READ2	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[9]	WRITE2	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[8]	EXEC2	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[7]	-	-
[6]	READ1	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[5]	WRITE1	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[4]	EXEC1	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[3]	-	-

Table 16-27. Primary Flash Permission Set Markers

Bit Position	Bit Field Name	Bit Description
[2]	READ0	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[1]	WRITE0	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[0]	EXEC0	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.

Figure 16-22. Secondary Flash Permission Set Markers

TCSDRn_SFPS0M0																															
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**
-	READ7	WRITE7	EXEC7	-	READ6	WRITE6	EXEC6	-	READ5	WRITE5	EXEC5	-	READ4	WRITE4	EXEC4	-	READ3	WRITE3	EXEC3	-	READ2	WRITE2	EXEC2	-	READ1	WRITE1	EXEC1	-	READ0	WRITE0	EXEC0
-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-28. Secondary Flash Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[31]	-	-
[30]	READ7	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[29]	WRITE7	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[28]	EXEC7	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[27]	-	-
[26]	READ6	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.

Table 16-28. Secondary Flash Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[25]	WRITE6	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[24]	EXEC6	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[23]	-	-
[22]	READ5	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[21]	WRITE5	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.

Table 16-28. Secondary Flash Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[20]	EXEC5	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[19]	-	-
[18]	READ4	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[17]	WRITE4	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[16]	EXEC4	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[15]	-	-

Table 16-28. Secondary Flash Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[14]	READ3	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[13]	WRITE3	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[12]	EXEC3	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[11]	-	-
[10]	READ2	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.

Table 16-28. Secondary Flash Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[9]	WRITE2	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[8]	EXEC2	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[7]	-	-
[6]	READ1	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[5]	WRITE1	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.

Table 16-28. Secondary Flash Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[4]	EXEC1	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.
[3]	-	-
[2]	READ0	READ READ grants reading permission for sectors 0 to 127. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response., '1': Reading this sector is allowed., If TCPEN is set, the Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on set.
[1]	WRITE0	WRITE WRITE grants writing and erasing permission for sectors 0 to 127. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing this sector is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively depending on the set.
[0]	EXEC0	Executing Code EXEC grants executing code in the sectors 0 to 127. '0': Executing code in this sector is forbidden and will cause bus error response. '1': Executing code is allowed. If TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCTCP0S and SCTCP1S respectively, depending on the set.

Note:

- The column bit position does not consider sector interleaving.
- On CY9DF126 (Atlas) the Flash permission set marker TCSDR0_FPS0M1-15 and TCSDR0_FPS1M1-15 are ignored by the hardware. Permissions of large interleaved sectors of both TCFLASH0 and TCFLASH1 are set by TCSDR1_FPS0M1-15 and TCSDR0_FPS1M1-15. Small sectors are still separately set per Flash macros.

- TCSDR1_FPKM is ignored since there is only one permission key supported for all Flash macros.
- The bit description for Set 1 Markers (TCSDRn_PFPS1M0~15/TCSDRn_SFPS1M0~15) are the same as for TCSDRn_PFPS0M0/TCSDRn_SFPS0M0.

16.2.14 EEFLASH Link Key Marker (EESDR0_LKM0~3)

The Link Key consists of 128 bits and is formed by four markers words. It is stored to each Flash macro. This marker holds the key for securing Flash macros for the case of erasing them. Unlike the security keys, this key is not configured for later transfer by external tool or by user program. The Link Key is programmed to Flash and read by Boot ROM only. See 16.2.11 TCFLASH Primary Link Key Marker (TCSDRn_PLKM0~3)/ TCFLASH Secondary Link Key Marker (TCSDRn_SLKM0~3) for further details.

EEFLASH Link Key Marker (EESDR0_LKM0~3)

Figure 16-23. EEFLASH Link Key Marker

EESDR_LKM0																															
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**
LKM[31]	LKM[30]	LKM[29]	LKM[28]	LKM[27]	LKM[26]	LKM[25]	LKM[24]	LKM[23]	LKM[22]	LKM[21]	LKM[20]	LKM[19]	LKM[18]	LKM[17]	LKM[16]	LKM[15]	LKM[14]	LKM[13]	LKM[12]	LKM[11]	LKM[10]	LKM[9]	LKM[8]	LKM[7]	LKM[6]	LKM[5]	LKM[4]	LKM[3]	LKM[2]	LKM[1]	LKM[0]
RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-29. EEFLASH Link Key Marker bits

Bit Position	Bit Field Name	Bit Description
[31:0]	LKM	Link Key Marker This marker holds the key for connecting the security of multiple Flash macros. = TCSDR0_LKM: If another macro other than TCFLASH0 has the same LKM value as TCFLASH0, then its code can be executed. != TCSDR0_LKM: If another macro other than TCFLASH0 has a different LKM value to TCFLASH0, its code cannot be executed. There is no special key value with a dedicated meaning.

Note: This marker is stored in separated words.

16.2.15 EEFLASH Permission Key Marker (EESDR_FPKM0~3)

The Flash Permission Key Marker is formed by four markers words. It consists of 127 bits key value. The marker is stored to each Flash macro. Refer to 16.2.12 TCFLASH Primary Permission Key Marker (TCSDRn_PFPKM0~3)/TCFLASH Secondary Permission Key Marker (TCSDRn_SFPKM0~3) for further details.

EEFLASH Permission Key Marker (EESDR_FPKM0~3)

Figure 16-24. EEFLASH Permission Key

EESDR_FPKM0																															
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
FPKM[31]	FPKM[30]	FPKM[29]	FPKM[28]	FPKM[27]	FPKM[26]	FPKM[25]	FPKM[24]	FPKM[23]	FPKM[22]	FPKM[21]	FPKM[20]	FPKM[19]	FPKM[18]	FPKM[17]	FPKM[16]	FPKM[15]	FPKM[14]	FPKM[13]	FPKM[12]	FPKM[11]	FPKM[10]	FPKM[9]	FPKM[8]	FPKM[7]	FPKM[6]	FPKM[5]	FPKM[4]	FPKM[3]	FPKM[2]	FPKM[1]	FPKM[0]
RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-30. EEFLASH Permission Key bits

Bit Position	Bit Field Name	Bit Description
[31:0]	FPKM	Flash Permission Key Marker These bits hold the 127 bits of the key. They correspond to the bits of registers SCTCPUSRKEY and SCEEPUSRKEY. These bits hold the secret key for switching permission, which is set and known only to the legal user software. There is no special key value with a dedicated meaning.

16.2.16 EEFLASH Permission Set Marker (EESDR_FPS0M0~3 and EESDR_FPS1M0~3)

A EEFLASH Permission Set Marker is formed by one marker word. It consists of 8 groups of three bits. Two of these markers are stored in each Flash macro.

This marker holds a collection of sector-wise flags to enable reading, execution and writing (erasing) of the corresponding sector. Two of these sets are provided for each macro, which can change the permission profile during run time. The alernative set can be selected by bit 127 of the register SCEEPUSRKEY.

Flash Permission Set Markers are processed independently of actual security setting in MSDR_PDSEM/MSDR_SDSEM but can be controlled by the Marker MSDR_PCTRLM/ MSDR_SCTRLM. All Flash Permission Set Markers stored in an EEFLASH macro which can be enabled or disabled by MSDR_PCRTLM:EEPEN and MSDR_SCRTLM:EEPEN respectively.

If Flash Permissions are enabled by EEPEN, the Boot ROM will copy the Flash Permission Key and the Flash Permision sets to the Security checker and activate the set as configured in Flash Permission Key Marker.

If it is disabled, Boot ROM will not configure permission keys and sets, instead leaving that to the user software.

EEFLASH Permission Set 0 Marker (EESDR_FPS0M0~3)

Figure 16-25. EEFLASH Permission Set Markers

EESDR_FPS0M0																															
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**
-	READ7	WRITE7	EXEC7	-	READ6	WRITE6	EXEC6	-	READ5	WRITE5	EXEC5	-	READ4	WRITE4	EXEC4	-	READ3	WRITE3	EXEC3	-	READ2	WRITE2	EXEC2	-	READ1	WRITE1	EXEC1	-	READ0	WRITE0	EXEC0
-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf	-	RfdlWf	RfdlWf	RfdlWf
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[31]	-	-
[30]	READ7	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[29]	WRITE7	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[28]	EXEC7	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[27]	-	-
[26]	READ6	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[25]	WRITE6	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[24]	EXEC6	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[23]	-	-
[22]	READ5	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[21]	WRITE5	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[20]	EXEC5	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[19]	-	-

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[18]	READ4	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[17]	WRITE4	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[16]	EXEC4	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[15]	-	-
[14]	READ3	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[13]	WRITE3	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[12]	EXEC3	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[11]	-	-
[10]	READ2	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[9]	WRITE2	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[8]	EXEC2	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[7]	-	-
[6]	READ1	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[5]	WRITE1	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[4]	EXEC1	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[3]	-	-

Table 16-31. EEFLASH Permission Set Markers bits

Bit Position	Bit Field Name	Bit Description
[2]	READ0	READ READ grants reading permission for sectors 0 to 31. This includes user symbols as constants or literal pool data even if used for code only. '0': Reading this sector is forbidden and will cause a bus error response. '1': Reading this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[1]	WRITE0	WRITE WRITE grants writing and erasing permission for sectors 0 to 31. '0': Writing to this sector is forbidden and will cause a bus error response. '1': Writing to this sector is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.
[0]	EXEC0	Executing Code EXEC grants executing code in the sectors 0 to 31. '0': Executing code in this sector is forbidden and will cause a bus error response. '1': Executing code is allowed. If MSDR_PCTRLM:TCPEN/MSDR_SCTRLM:TCPEN is set, Boot ROM copies the marker to the Security Checker registers SCCFG_SCTCP0S and SCCFG_SCTCP1S respectively depending on set.

- The bit descriptions for Set 1 Markers EESDR_FPS1M0~15 are the same as for EESDR_FPS0M0 r

16.3 SDR Validation Status

After each reset which causes Boot ROM to perform security check, the user configuration has to be read from SDR in Flash. The validation result is provided in register SCCFG_GPREG1.

Purpose of SDR Validation Status

Validation status can be used by operator to identify problems which block user from unlocking secured device despite using correct key or which cause code in interleaved Flash macros not being executable.

- User boot loader operation failed
- JTAG programming failed
- ECC double error occured in security configuration

Such failures could result in selection of wrong SDR configuration set or even lock device with no means to unlock it.

The validation status does not provide a backdoor to device security. It allows limited diagnostics to narrow a failure to external tooling, application software or hardware. It doesnot allow any feedback to Boot ROM and its behaviour.

Shared Usage of SCCFG_GPREG1

The status is written by Boot ROM to SCCFG_GPREG1. This register is not specific to Boot ROM or reserved for it. If JTAG operator or user program has to use GPREG1, it can be overwritten at any time,Optionally the validation status can be saved before.

SCCFG_GPREG1 is used, it can be read via JTAG even the device is secured and locked.

Markers covered by status Information

The status information covers the selection of either primary or secondary configuration in Main SDR (MSDR) which includes correctness of Configuration Enable Markers (MSDR_PCEM or MSDR_SCEM) but also correctness of ECC for other MSDR markers. For details on MSDR selection refer to 14a.4 I Primary and Secondary Device Security Configuration.

Status also covers correctness of ECC of permissions keys and Permissions sets in Extended SDR (ESDR). ESDR consists of TCSDR<n> respectively. EESDR<n> for each Flash macro.

TCFLASH Permission Key (TCSDR0 only) and Permission sets (all TSCSDR<n>) are valid only if read completely without double bit ECC error. Otherwise, all TCFLASH Permission Configuration is disabled regardless of setting in MSDR_PCTRLM or MSDR_SCTRLM.

Link keys in ESDR are not included in validation status.

16.3.1 SDR Validation Status in GPREG1 (SCCFG_GPREG1)

The SCCFG_GPREG1 register is not specific to Boot ROM nor reserved for it. If JTAG operator or user program has to use GPREG1, it can be overwritten at any time. Optionally the validation status can be saved before SCCFG_GPREG1 is used and it can be read via JTAG even when the device is secured and locked.

SDR Validation Status in GPREG1 (SCCFG_GPREG1)

Figure 16-26. SDR Validation Status

SCCFG_GPREG1																															
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
ECCERR	MSDRSEL[2]	MSDRSEL[1]	MSDRSEL[0]	TCFPST[1]	TCFPST[0]	EEFPST[1]	EEFPST[0]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

There are three default values:

- The GPREG1 value is 0x00000000 after a hard reset before Boot ROM execution.
- The GPREG1 value is 0x45000000 after a hard reset and Boot ROM execution with fully erased MSDR (factory configuration). Other SDR configurations may result in different values.
- GPREG1 is undefined after any reset which does not reset the Security Checker because any value written by the user before is preserved.

Table 16-32. SDR Validation Status bits

Bit Position	Bit Field Name	Bit Description
[31]	ECCERR	ECC Error While reading SDR, a double-bit ECC error occurred. This may indicate a possible cause for the device being permanently locked or the incorrect MSDR/TCSDR0 shadow being selected.

Table 16-32. SDR Validation Status bits

Bit Position	Bit Field Name	Bit Description
[30:28]	MSDRSEL	Validate MSDR MSDRSEL shows result of MSDR validation. '000': Reset value '001': Reserved '010': Primary MSDR configuration selected '011': Secondary MSDR configuration selected '100': Blank primary MSDR selected (all 0xFFFFFFFF with no double-bit ECC error) '101': Both primary and secondary MSDR are invalid '110': Reserved '111': Reserved Values '010', '011' and '100' indicate at least one valid configuration. Value '101' indicates two invalid configurations. For details on valid and invalid configurations see relevant marker description. Note: This selection also controls selection of the primary or secondary configuration of TCSDR0.
[27:26]	TCFPST	Validate TCFLASH Permission TCFPST shows the result of the validation of Flash Permissions for all TCFLASH macros. It covers double-bit ECC errors of the Permission Key (TCSDR0 only) and Permissions sets (all TCFLASH macros). If there is an ECC error, the permission configuration is invalid. This implies that permission is set off in User Mode. '00': Reset value '01': TCFLASH Permission is enabled '10': TCFLASH Permission is disabled '11': TCFLASH Permission is invalid and disabled
[25:24]	EEFPST	Validate EEFLASH Permission EEFPST shows the result of the validation of Flash Permissions for all EEFLASH macros. It covers double-bit ECC errors of the Permission Key (EESDR0 only) and Permissions sets (all EEFLASH macros). If there is an ECC error, the permission configuration is invalid. This implies that permission is set off in User Mode. '00': Reset value '01': EEFLASH Permission is enabled '10': EEFLASH Permission is disabled '11': EEFLASH Permission is invalid and disabled
[23:0]	-	-

Note:

- The selection of Primary and Secondary configuration of TCSDR0 is the same as MSDR selection and not separately shown.
- Permission key is read from TCSDR0 and EESDR0 only even if marker is provided by all Flash macros.
- Independently of Boot ROM which configures Permission sets for all Flash macros, the dedicated device may support, ignore or share Permission sets for dedicated Flash macros.

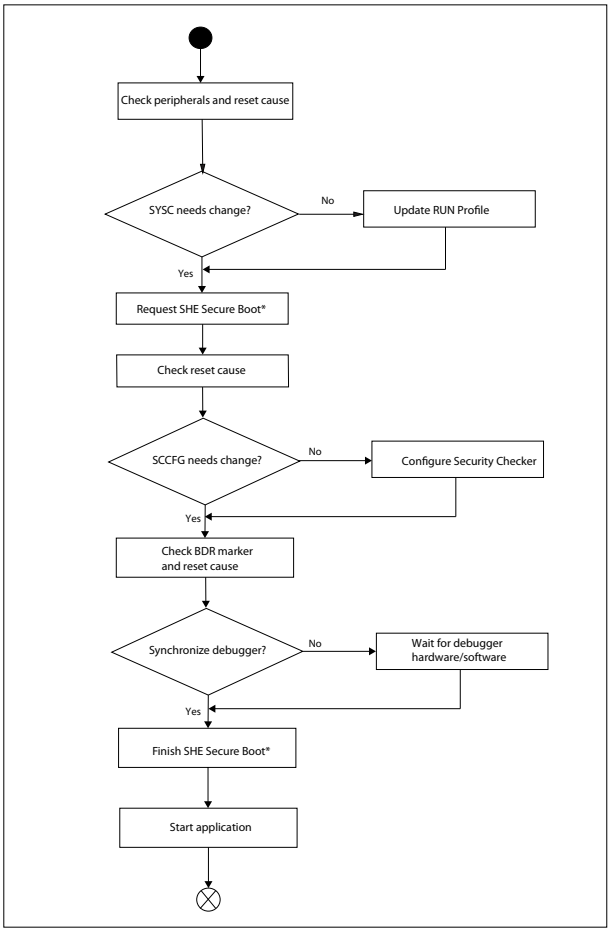
16.4 Operation of the Boot ROM

This section describes the operation of the Boot ROM

Booting of a device

On every reset, the Boot ROM is executed first. After the device has been configured, either the user application is started or test mode is activated, depending on the device mode selected by external pins.

Figure 16-27. Boot ROM embedded in a host system



*Only on devices equipped with the Secure Hardware Extension (SHE)

Configuration of System

Controller and other peripherals In order to ensure a reliable and secure environment for a later Boot ROM operation, several peripheral settings are checked. If these are not suitable, the configuration is changed. Some temporary settings (e.g. PPU) will be restored before the application starts and are not described here. Other settings are kept until the application starts/finishes/runs?? and may change a configuration previously selected by the application (e.g. before changing to Power Saving State (PSS)).

1. In the event of a hard reset, the configuration of the System Controller is not changed by the Boot ROM because the proper configuration is already applied by the hardware. A hard reset is indicated when any of the lower 10 bits of the reset cause register has a value of ’1’.
2. In the event of a reset of power domain 2 (PD2) instead of hard reset, the System Controller is configured. This condition is indicated when the reset cause bit (PD2R) is ’1’ and all lower 10 bits of the reset cause register are ’0’. Note: the reset cause bit FAKEPDR is ignored and the System Controller is checked even if only a fake power domain reset is applied.

a. The following settings are unconditionally changed on CY9DF126 Rev1 (“Atlas”)
- SYSC_SYSINTER:RUNDNIE is cleared to ’0’.
- SYSC_SYSSTSR:RUNDN is cleared to ’0’.

b. The following profile settings are changed if their status is different than given below. If any of the following settings is different, all the following settings are updated and this requires a profile update.
- SYSC_RUNCKDIVR0:SYSDIV is cleared to 0.
- SYSC_RUNCKER:ENTRACEPD2 is set to ’1’.
- SYSC_RUNCKER:ENTRACEPD3 is set to ’1’.
- SYSC_RUNCKDIVR0:HPMDIV is cleared to 0.
- SYSC_RUNCKDIVR0:CFGDIV is cleared to 0.
- SYSC_RUNCKSELR:PERI3CSL is cleared to 0.
- SYSC_RUNCKDIVR2:PERI3DIV is cleared to 0.
- SYSC_RUNCKER:ENPERI3PD2 is set to ’1’.
- SYSC_RUNLVDCFGR:LVDE50 is set to ’1’.
- SYSC_RUNLVDCFGR:LVDR50 is set to ’1’.
- SYSC_RUNLVDCFGR:SV50 is set to 3.
- SYSC_RUNLVDCFGR:LVDE12 is set to ’1’.
- SYSC_RUNLVDCFGR:LVDR12 is set to ’1’.
- SYSC_RUNLVDCFGR:SV12 is set to 2.

Secure Boot

A Secure Boot is an operation of the Secure Hardware Extension (SHE) which enables boot-protected keys stored inside SHE by the authentication of application boot code or part of it. Boot ROM handles Secure Boot only on devices with SHE.

Secure Boot is handled in two phases:

- In the first phase, Secure Boot is requested with proper parameters in advance. Boot ROM does not know whether Secure Boot is enabled inside SHE.
  a. The command code for CMD_SECURE_BOOT is written to the register SHE_CMD. The parameter Boot Size is read from the Flash marker BDR_SBSM and written to SHE input FIFO.
  b. The SHE AXI Channel master is configured and started according to the Secure Boot parameters ‘Boot Size’ and ‘Boot Start’ in order to copy the application boot code to SHE with high throughput.
  - Boot Size is read from marker BDR_SBSM. If the byte count stored in BDR_SBSM is not a multiple of 16 bytes (128 bits), Boot ROM increases it to next higher multiple of 128 bit. This ensures proper padding of the data transferred to FIFO while the MAC calculation still considers the original byte count stored in BDR_SBSM.

  Note:

  The SHE AXI Channel Master may access memory beyond the area spanned by the byte count. If it causes a bus error on access, Secure Boot authentication of code will fail and boot-protected keys are locked.

  - Boot Start is read from marker BDR_ABVM if it is enabled by BDR_AVBEM. If BDR_ABVM is disabled by BDR_ABVEM, Boot Start is fixed to the TCM address 0x80_0000 resp. AXI address 0x0100_0000.

  Note:

  The Boot ROM converts the TCM address of BDR_ABVM into AXI address as needed by SHE AXI Channel Master.

  Since the AXI Channel Master operates on 64-bit words, Boot ROM will align the start address to the next lower 64-bit address. If the user-configured address is not aligned, but an off-chip calculated MAC is stored to SHE, this has to be considered.

  If Secure Boot MAC is generated by SHE self-learning mode, alignment of start address will apply to both generation and verification. However, increased Secure Boot Size has to be considered to ensure that shifted address range still includes the proper end.

  Note:

  An 128-bit aligned start address is recommended to enable the SHE AXI Channel Master to transfer bursts more effectively.

c. In the event of any reset which did not reset the Security Checker, the Boot ROM enables debugging if enabled by Device Security.

SHE will execute Secure Boot autonomously and concurrently to another Boot ROM operation.

In a second phase after checking Device Security, the Boot ROM will synchronize to SHE and finish Secure Boot.

d. Boot ROM waits until SHE is up and running (SHE_STATUS:INITDONE).

e. Depending on Secure Boot Mode, the Boot ROM either waits for the Secure Boot to complete before starting the application or it starts the application immediately:

- Secure Boot Mode is read from marker BDR_SBMM.

The BDR_SBMM value 0x0000_0000 selects the immediate start of application. Secure Boot may run in the background.

BDR_SBMM other than 0x0000_0000 selects the Boot ROM to wait for Secure Boot to finish.

Note:

The blank (erased) BDR of the new MCU selects the Boot ROM to wait. However, there is no effective waiting time because Secure Boot is disabled inside SHE and Secure Boot will complete after SHE initialization.

If a proper Secure Boot request fails (e.g. due to an ECC error when reading parameters from BDR) in the first Secure Boot phase, the Boot ROM will request a dummy Secure Boot to enforce SHE boot failure.

If Secure Boot stops with a command error, the Boot ROM will attempt to unlock the SHE AXI Channel Master using the command “cancel request” in its second Secure Boot phase.

Security Checker Configuration

Device Security and Flash Security refers to the read-out Flash protection via JTAG and the CPU access protection respectively.

The user configuration is stored in the SDRs in Flash memory. The Boot ROM reads this configuration and adjusts the Security Checker accordingly.

a. The Security Checker needs to be configured only if its power domain was reset. Otherwise, it is locked against reconfiguration. The Boot ROM checks the reset cause register for a hard reset cause (lower 10 bits of the reset cause register) or Power Domain 2 Reset (PD2R). However, with PD2R set with the concurrent FAKEPDR but without any hard reset bit is treated as a fake reset which does not require nor allow reconfiguration.

b. If reset cause requires configuration of the Security Checker, the Main Security Description Record (MSDR) is processed first, and either the primary or secondary shadow is selected. (see Primary and Secondary Device Security Configuration).

c. The Security Checker is configured according to the selected primary or secondary shadow of the MSDR.

d. The Link Keys in the Extended SDR of different Flash macros are compared against the Link Key of the Flash macro which contains the MSDR. If the Link Key of another macro does not match the reference, both code execution in this Flash macro and reading data from this macro are disabled.

e. Permission Sets are copied from the Extended SDR to the Security Checker for each macro.

External Debugger Synchronization

The external debug tool may have to configure the debug unit before the application is started. However, set-up of the connection and configuration of the debug unit may take longer than normal Boot ROM execution.

In order to be able to debug from the first instruction in the application, the start of the application is suspended until the debugger is properly connected. This feature can be disabled by BDR_DWEM.

a. The Boot ROM checks BDR_DWEM. If waiting is disabled, the Boot ROM skips further waiting on the debugger. Applications in the production stage are those most likely to disable debugger synchronization. Waiting can be re-enabled by writing the proper value to the marker. Disabling debugger synchronization again is not possible but requires erasing the sector with BDR.

b. If waiting is enabled, the Boot ROM checks the reset case. Debug configuration is only reset with Power On Reset (PRSTX) and External Reset (RSTX).
In any other case, debug configuration is preserved and there is no need to synchronize. Further waiting is skipped. If waiting is still required, two phases are still available.

c. During a first waiting period, the presence of the JTAG hardware tool is tested. If no JTAG tool is found after 19,200 RC-clock cycles (+5%), further waiting is skipped and the application is started. In this case, the delay is about 0.8 ms to 3.3 ms at 24 MHz to 6 MHz RC-clock.

d. If the JTAG tool is found, a second waiting period begins to accommodate the reception of an acknowledgement from the host application (debugger). If one is not received, the waiting period is cancelled after "2^{23} " RC-clock cycles. In this case, the delay is about 350 ms to "1.4 s" at 24 MHz to 6 MHz RC-clock.
The acknowledgement informs the MCU that the host application has transferred all necessary data (e.g. security keys, breakpoint configuration) and is ready to execute the user program.
If tooling is used, the actual delay until connection depends on the external JTAG clock frequency and athe mount of data to be transferred.

Starting the Application

Jumping to the application is the last operation by the Boot ROM.

The start address is the same as that used for the SHE Secure Boot Start address.

If enabled by BDR_ABVEM, BDR_ABVM is used. Otherwise or if there is an ECC error in these markers, the application is started at a fixed TCM address 0x00800000.

a. Before the application is started, CPU registers R0 to R13 are cleared to 0. Only the Link register R14 will contain a non-zero value as it holds the application start address.

b. The application is jumped to in Arm state. Any Thumb state indicating bit 0 in BDR_ABVM is ignored.

Primary and Secondary Device Security Configuration

The concept of two alternative security configurations in different Flash sectors secures the device with one configuration while the other is erased.

During normal operation, the secondary shadow of MSDR could be kept blank or a copy of the primary MSDR could serve as a fall-back for defects. The interleaved sectors of MSDR are best for the boot loader part of the application, which is updated rarely. If, however, an update is necessary, the boot loader part of the application could utilize the secondary shadow to preserve a valid MSDR configuration even if both sectors with MSDR are separately erased.

The active configuration is selected by the Boot ROM during device booting. The actual selection can be checked by the application or by an external operator in the General Purpose Register (SCCFG_GPREG1) after booting.

- The primary configuration is validated first. If enabled by PCEM and if no ECC error greater than 1-bit occurs while reading all MSDR markers, the primary configuration is valid and defines the security scheme. Otherwise, it is invalid.
- If the primary configuration is invalid, the secondary configuration is then validated. If enabled by SCEM and if no ECC error greater than 1-bit occurs while reading all MSDR markers, the secondary configuration is valid and defines the security scheme. Otherwise, it is invalid.
- If both configurations are invalid by PCEM and SCEM or by ECC errors, the primary configuration is checked to see if it is blank (erased). If all markers (PDSEM, PDSKM, PFFAKM, PCTRLM, PCEM) are read as 0xFFFFFFFF without any ECC error greater than 1-bit, this blank configuration defines the security scheme as unsecured.

Table 16-33. Primary and Secondary Device Security Configuration

Condition No.	Primary MSDR		Secondary MSDR		Selection
	Marker	ECC Error	Marker	ECC Error	
1	PCEM == 0x292D3A7B	No	-	-	Primary
2	All markers == 0xFFFFFFFF	No	SCEM != 0x292D3A7B	No	Primary (blank)
3			-	Yes	

Table 16-33. Primary and Secondary Device Security Configuration

Condition No.	Primary MSDR		Secondary MSDR		Selection
	Marker	ECC Error	Marker	ECC Error	
4	PCEM != 0x292D3A7B	No	SCEM == 0x292D3A7B	No	Secondary
5	-	Yes			
6	Any markers != 0xFFFFFFFF && PCEM != 0x292D3A7B	No	SCEM != 0x292D3A7B	No	Invalid MSDR
7			-	Yes	
8	-	Yes	SCEM != 0x292D3A7B	No	
9				Yes	

Table 16-33 shows the different combinations of settings for Primary MSDR and Secondary MSDR.

- Primary selection means that all MSDR settings are read from the Primary MSDR.
- Secondary selection means that all MSDR settings are read from the Secondary MSDR.
- An invalid MSDR locks the device against any user key transfer.

ECC error means detected 2-bit error (or higher) in any read marker of that MSDR.

RAM Usage

Boot ROM needs to use RAM for its operation. This RAM might be overwritten on start-up and is not restored after Boot ROM operation. Any contents written by the application before reset is not preserved.The RAM used by Boot ROM before starting the application can be used by the application. However, it should not allocate data which have to be preserved beyond reset.

Besides other data, Boot ROM also locates stacks in this area. However, only the stack DABORT is kept until properly configured by the application. Imprecise DABORT is enabled by Boot ROM.

The RAM used by Boot ROM is 512 Bytes located at the end of the TCM RAM. The absolute addresses depend on the size of the TCM RAM:

- End of 64 KB TCM RAM: 0x0000FE00..0x0000FFFF
- End of 128 KB TCM RAM: 0x0001FE00..0x0001FFFF

Register Usage

Boot ROM needs to use general purpose registers for its operation. However, all registers R0...R13 of the SVC mode are cleared before starting the application. This includes the stack pointer R13. However, link register R14 may contain the start address of the application.

17. Interrupt Controller

This chapter explains the functions and operations of the Interrupt Controller (IUNIT).

17.1 Outline of Interrupt Controller

This section describes the features and the block diagram of the Interrupt Controller.

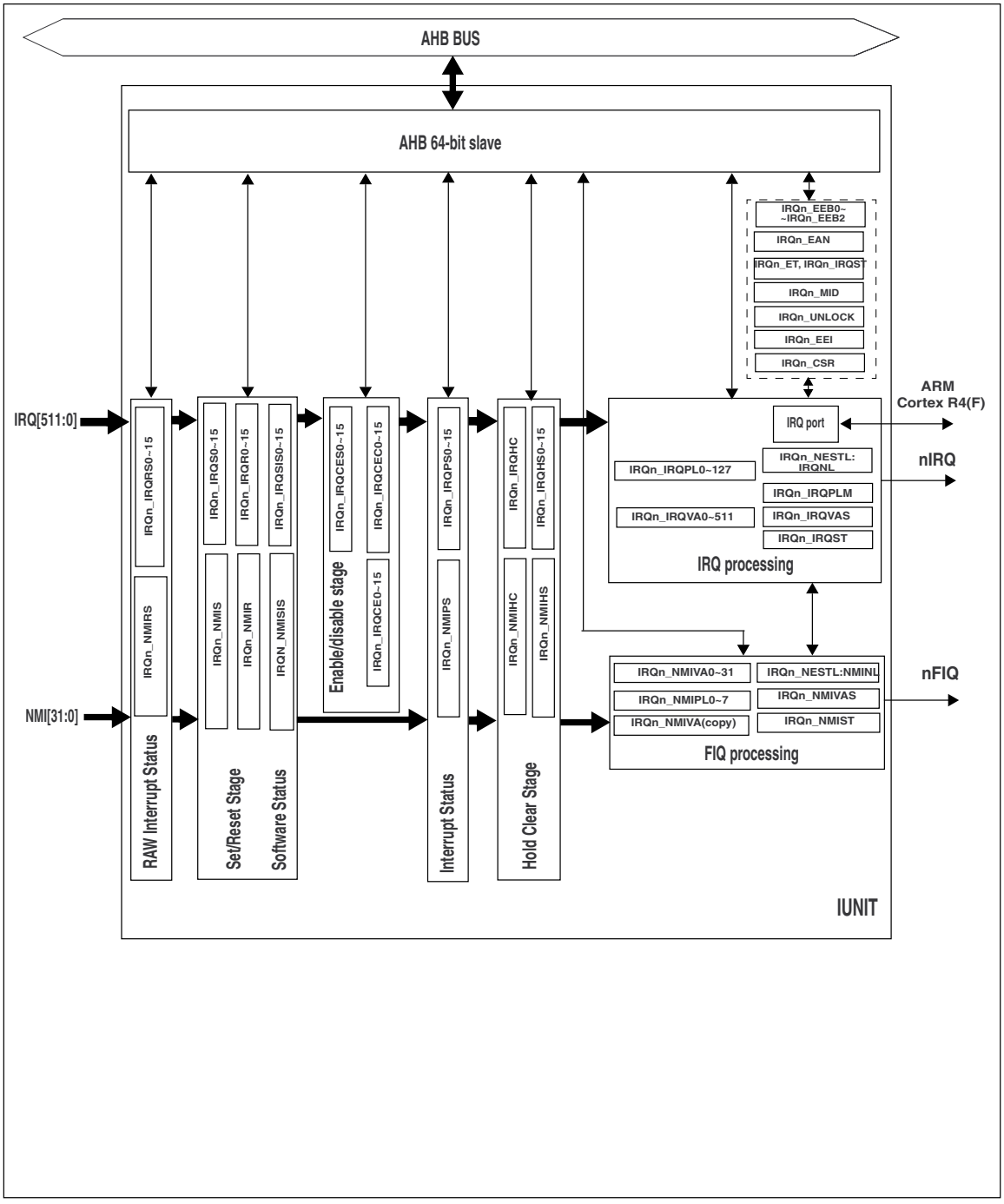
Features of Interrupt Controller

Interrupt Controller supports 512 Normal Interrupt (IRQ) sources and 32 Non-Maskable Interrupt (NMI) sources. All IRQ sources are processed to produce a single Interrupt Request (nIRQ) to the processor. The IRQ interrupt supports software interrupts, priority filtering, and vector generation. The IRQ interrupts can be masked. All NMI sources are processed to produce a single Fast Interrupt Request (nFIQ) interrupt to the processor. The NMI interrupt supports software interrupts, programmable software priority, and vector generation. The features of Interrupt Controller are as follows:

- Controls the Interrupt Lines (nIRQ and nFIQ) of an Arm Cortex-R4(F) processor
- Supports 32 NMI source for nFIQ generation
- Supports 512 Normal Interrupt sources for nIRQ generation
- Default hardware interrupts priority levels.
- Interrupts mapped to lower order line sources have higher priority
- Supports request for low power mode from the System Controller
- Internal synchronization of asynchronous interrupt sources
- Programmable 32-level priority controller for normal IRQ sources. Also, supports programmable priority level masking
- Programmable 16-level priority controller for NMI interrupt sources
- Supports vectored interrupts for both IRQ and NMI
- NMI source 0 has fixed priority set to '0'
- One 32-bit vectored address per interrupt source
- Software interrupt generation
- Privileged mode support for restricted access

Block diagram of IUNIT

Figure 17-1. Block diagram of IUNIT



17.2 Interrupt Controller registers

This section describes the registers of the IUNIT in detail. IUNIT can be programmed and monitored through its Configuration and Status Registers.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of IUNIT

The following registers are available for IUNIT module:

- IUNIT NMI Vector Address Status Register (IRQn_NMIVAS)
- IUNIT NMI Status Register (IRQn_NMIST)
- IUNIT IRQ Vector Address Status Register (IRQn_IRQVAS)
- IUNIT IRQ Status Register (IRQn_IRQST)
- IUNIT NMI Vector Address Register (IRQn_NMIVA0~31)
- IUNIT IRQ Vector Address Register (IRQn_IRQVA0~511)
- IUNIT NMI Priority Level Register (IRQn_NMIPL0)
- IUNIT NMI Priority Level Register (IRQn_NMIPL1~7)
- IUNIT IRQ Priority Level Register (IRQn_IRQPL0~127)
- IUNIT NMI Set Register (IRQn_NMIS)
- IUNIT NMI Reset Register (IRQn_NMIR)
- IUNIT NMI Software Interrupt Status Register (IRQn_NMISIS)
- IUNIT IRQ Set Register (IRQn_IRQS0~15)
- IUNIT IRQ Reset Register (IRQn_IRQR0~15)
- IUNIT IRQ Software Interrupt Status Register (IRQn_IRQSIS0~15)
- IUNIT IRQ Channel Enable Set Register (IRQn_IRQCES0~15)
- IUNIT IRQ Channel Enable Clear Register (IRQn_IRQCEC0~15)
- IUNIT IRQ Channel Enable Register (IRQn_IRQCE0~15)
- IUNIT NMI Hold Clear Register (IRQn_NMIHC)
- IUNIT NMI Hold Status Register (IRQn_NMIHS)
- IUNIT IRQ Hold Clear Register (IRQn_IRQHC)
- IUNIT IRQ Hold Status Register (IRQn_IRQHS0~15)
- IUNIT IRQ Priority Level Mask Register (IRQn_IRQPLM)
- IUNIT Control and Status Register (IRQn_CSR)
- IUNIT Nesting Level Status Register (IRQn_NESTL)
- IUNIT NMI Raw Status Register (IRQn_NMIRS)
- IUNIT NMI Preprocessed Status Register (IRQn_NMIPS)
- IUNIT IRQ Raw Status Register (IRQn_IRQRS0~15)
- IUNIT IRQ Preprocessed Status Register (IRQn_IRQPS0~15)
- IUNIT Unlock Register (IRQn_UNLOCK)
- IUNIT Module Identification Register (IRQn_MID)
- IUNIT ECC Error Interrupt Register (IRQn_EEI)
- IUNIT ECC Address Number Register (IRQn_EAN)
- IUNIT ECC Test Register (IRQn_ET)
- IUNIT ECC Error Bits Register (IRQn_EEB0 ~ 1)
- IUNIT ECC Error Bits Register (IRQn_EEB2)

Memory layout of IUNIT registers

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	IRQn_NMIST 00000000 00000000 00001111 00100000				IRQn_NMIVAS 00000000 00000000 00000000 00000000			
0x00000008	IRQn_IRQST 00000000 00011111 00000010 00000000				IRQn_IRQVAS 00000000 00000000 00000000 00000000			
0x00000010	IRQn_NMIVA1 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA0 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000018	IRQn_NMIVA3 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA2 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000020	IRQn_NMIVA5 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA4 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000028	IRQn_NMIVA7 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA6 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000030	IRQn_NMIVA9 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA8 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000038	IRQn_NMIVA11 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA10 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000040	IRQn_NMIVA13 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA12 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000048	IRQn_NMIVA15 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA14 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000050	IRQn_NMIVA17 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA16 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000058	IRQn_NMIVA19 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA18 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000060	IRQn_NMIVA21 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA20 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000068	IRQn_NMIVA23 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA22 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000070	IRQn_NMIVA25 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA24 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000078	IRQn_NMIVA27 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA26 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000080	IRQn_NMIVA29 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA28 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000088	IRQn_NMIVA31 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_NMIVA30 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000090	IRQn_IRQVA1 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA0 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000098	IRQn_IRQVA3 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA2 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000A0	IRQn_IRQVA5 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA4 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000A8	IRQn_IRQVA7 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA6 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000B0	IRQn_IRQVA9 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA8 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000B8	IRQn_IRQVA11 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA10 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000C0	IRQn_IRQVA13 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA12 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000C8	IRQn_IRQVA15 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA14 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000D0	IRQn_IRQVA17 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA16 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000D8	IRQn_IRQVA19 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA18 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000E0	IRQn_IRQVA21 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA20 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000E8	IRQn_IRQVA23 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA22 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000F0	IRQn_IRQVA25 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA24 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000000F8	IRQn_IRQVA27 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA26 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000100	IRQn_IRQVA29 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA28 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000108	IRQn_IRQVA31 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA30 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000110	IRQn_IRQVA33 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA32 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000118	IRQn_IRQVA35 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA34 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000120	IRQn_IRQVA37 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA36 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000128	IRQn_IRQVA39 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA38 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000130	IRQn_IRQVA41 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA40 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000138	IRQn_IRQVA43 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA42 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000140	IRQn_IRQVA45 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA44 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000148	IRQn_IRQVA47 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA46 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000150	IRQn_IRQVA49 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA48 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000158	IRQn_IRQVA51 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA50 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000160	IRQn_IRQVA53 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA52 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000168	IRQn_IRQVA55 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA54 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000170	IRQn_IRQVA57 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA56 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000178	IRQn_IRQVA59 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA58 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000180	IRQn_IRQVA61 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA60 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000188	IRQn_IRQVA63 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA62 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000190	IRQn_IRQVA65 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA64 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000198	IRQn_IRQVA67 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA66 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001A0	IRQn_IRQVA69 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA68 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001A8	IRQn_IRQVA71 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA70 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001B0	IRQn_IRQVA73 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA72 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001B8	IRQn_IRQVA75 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA74 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000001C0	IRQn_IRQVA77 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA76 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001C8	IRQn_IRQVA79 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA78 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001D0	IRQn_IRQVA81 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA80 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001D8	IRQn_IRQVA83 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA82 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001E0	IRQn_IRQVA85 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA84 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001E8	IRQn_IRQVA87 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA86 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001F0	IRQn_IRQVA89 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA88 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000001F8	IRQn_IRQVA91 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA90 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000200	IRQn_IRQVA93 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA92 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000208	IRQn_IRQVA95 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA94 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000210	IRQn_IRQVA97 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA96 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000218	IRQn_IRQVA99 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA98 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000220	IRQn_IRQVA101 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA100 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000228	IRQn_IRQVA103 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA102 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000230	IRQn_IRQVA105 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA104 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000238	IRQn_IRQVA107 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA106 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000240	IRQn_IRQVA109 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA108 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000248	IRQn_IRQVA111 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA110 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000250	IRQn_IRQVA113 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA112 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000258	IRQn_IRQVA115 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA114 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000260	IRQn_IRQVA117 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA116 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000268	IRQn_IRQVA119 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA118 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000270	IRQn_IRQVA121 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA120 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000278	IRQn_IRQVA123 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA122 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000280	IRQn_IRQVA125 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA124 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000288	IRQn_IRQVA127 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA126 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000290	IRQn_IRQVA129 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA128 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000298	IRQn_IRQVA131 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA130 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002A0	IRQn_IRQVA133 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA132 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002A8	IRQn_IRQVA135 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA134 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002B0	IRQn_IRQVA137 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA136 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002B8	IRQn_IRQVA139 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA138 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002C0	IRQn_IRQVA141 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA140 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002C8	IRQn_IRQVA143 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA142 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002D0	IRQn_IRQVA145 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA144 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002D8	IRQn_IRQVA147 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA146 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002E0	IRQn_IRQVA149 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA148 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002E8	IRQn_IRQVA151 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA150 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000002F0	IRQn_IRQVA153 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA152 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000002F8	IRQn_IRQVA155 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA154 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000300	IRQn_IRQVA157 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA156 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000308	IRQn_IRQVA159 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA158 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000310	IRQn_IRQVA161 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA160 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000318	IRQn_IRQVA163 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA162 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000320	IRQn_IRQVA165 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA164 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000328	IRQn_IRQVA167 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA166 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000330	IRQn_IRQVA169 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA168 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000338	IRQn_IRQVA171 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA170 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000340	IRQn_IRQVA173 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA172 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000348	IRQn_IRQVA175 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA174 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000350	IRQn_IRQVA177 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA176 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000358	IRQn_IRQVA179 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA178 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000360	IRQn_IRQVA181 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA180 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000368	IRQn_IRQVA183 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA182 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000370	IRQn_IRQVA185 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA184 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000378	IRQn_IRQVA187 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA186 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000380	IRQn_IRQVA189 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA188 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000388	IRQn_IRQVA191 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA190 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000390	IRQn_IRQVA193 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA192 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000398	IRQn_IRQVA195 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA194 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003A0	IRQn_IRQVA197 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA196 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003A8	IRQn_IRQVA199 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA198 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003B0	IRQn_IRQVA201 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA200 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003B8	IRQn_IRQVA203 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA202 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003C0	IRQn_IRQVA205 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA204 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003C8	IRQn_IRQVA207 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA206 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003D0	IRQn_IRQVA209 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA208 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003D8	IRQn_IRQVA211 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA210 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003E0	IRQn_IRQVA213 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA212 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003E8	IRQn_IRQVA215 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA214 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003F0	IRQn_IRQVA217 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA216 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000003F8	IRQn_IRQVA219 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA218 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000400	IRQn_IRQVA221 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA220 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000408	IRQn_IRQVA223 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA222 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000410	IRQn_IRQVA225 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA224 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000418	IRQn_IRQVA227 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA226 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000420	IRQn_IRQVA229 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA228 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000428	IRQn_IRQVA231 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA230 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000430	IRQn_IRQVA233 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA232 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000438	IRQn_IRQVA235 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA234 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000440	IRQn_IRQVA237 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA236 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000448	IRQn_IRQVA239 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA238 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000450	IRQn_IRQVA241 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA240 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000458	IRQn_IRQVA243 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA242 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000460	IRQn_IRQVA245 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA244 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000468	IRQn_IRQVA247 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA246 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000470	IRQn_IRQVA249 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA248 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000478	IRQn_IRQVA251 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA250 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000480	IRQn_IRQVA253 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA252 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000488	IRQn_IRQVA255 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA254 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000490	IRQn_IRQVA257 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA256 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000498	IRQn_IRQVA259 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA258 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004A0	IRQn_IRQVA261 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA260 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004A8	IRQn_IRQVA263 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA262 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004B0	IRQn_IRQVA265 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA264 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000004B8	IRQn_IRQVA267 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA266 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004C0	IRQn_IRQVA269 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA268 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004C8	IRQn_IRQVA271 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA270 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004D0	IRQn_IRQVA273 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA272 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004D8	IRQn_IRQVA275 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA274 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004E0	IRQn_IRQVA277 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA276 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004E8	IRQn_IRQVA279 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA278 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004F0	IRQn_IRQVA281 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA280 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000004F8	IRQn_IRQVA283 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA282 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000500	IRQn_IRQVA285 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA284 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000508	IRQn_IRQVA287 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA286 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000510	IRQn_IRQVA289 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA288 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000518	IRQn_IRQVA291 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA290 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000520	IRQn_IRQVA293 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA292 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000528	IRQn_IRQVA295 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA294 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000530	IRQn_IRQVA297 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA296 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000538	IRQn_IRQVA299 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA298 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000540	IRQn_IRQVA301 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA300 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000548	IRQn_IRQVA303 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA302 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000550	IRQn_IRQVA305 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA304 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000558	IRQn_IRQVA307 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA306 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000560	IRQn_IRQVA309 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA308 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000568	IRQn_IRQVA311 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA310 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000570	IRQn_IRQVA313 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA312 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000578	IRQn_IRQVA315 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA314 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000580	IRQn_IRQVA317 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA316 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000588	IRQn_IRQVA319 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA318 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000590	IRQn_IRQVA321 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA320 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000598	IRQn_IRQVA323 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA322 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005A0	IRQn_IRQVA325 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA324 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005A8	IRQn_IRQVA327 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA326 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005B0	IRQn_IRQVA329 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA328 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005B8	IRQn_IRQVA331 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA330 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005C0	IRQn_IRQVA333 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA332 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005C8	IRQn_IRQVA335 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA334 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005D0	IRQn_IRQVA337 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA336 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005D8	IRQn_IRQVA339 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA338 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005E0	IRQn_IRQVA341 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA340 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000005E8	IRQn_IRQVA343 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA342 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005F0	IRQn_IRQVA345 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA344 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000005F8	IRQn_IRQVA347 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA346 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000600	IRQn_IRQVA349 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA348 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000608	IRQn_IRQVA351 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA350 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000610	IRQn_IRQVA353 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA352 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000618	IRQn_IRQVA355 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA354 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000620	IRQn_IRQVA357 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA356 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000628	IRQn_IRQVA359 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA358 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000630	IRQn_IRQVA361 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA360 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000638	IRQn_IRQVA363 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA362 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000640	IRQn_IRQVA365 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA364 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000648	IRQn_IRQVA367 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA366 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000650	IRQn_IRQVA369 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA368 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000658	IRQn_IRQVA371 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA370 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000660	IRQn_IRQVA373 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA372 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000668	IRQn_IRQVA375 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA374 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000670	IRQn_IRQVA377 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA376 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000678	IRQn_IRQVA379 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA378 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000680	IRQn_IRQVA381 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA380 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000688	IRQn_IRQVA383 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA382 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000690	IRQn_IRQVA385 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA384 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000698	IRQn_IRQVA387 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA386 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006A0	IRQn_IRQVA389 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA388 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006A8	IRQn_IRQVA391 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA390 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006B0	IRQn_IRQVA393 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA392 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006B8	IRQn_IRQVA395 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA394 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006C0	IRQn_IRQVA397 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA396 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006C8	IRQn_IRQVA399 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA398 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006D0	IRQn_IRQVA401 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA400 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006D8	IRQn_IRQVA403 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA402 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006E0	IRQn_IRQVA405 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA404 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006E8	IRQn_IRQVA407 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA406 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006F0	IRQn_IRQVA409 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA408 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000006F8	IRQn_IRQVA411 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA410 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000700	IRQn_IRQVA413 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA412 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000708	IRQn_IRQVA415 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA414 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000710	IRQn_IRQVA417 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA416 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000718	IRQn_IRQVA419 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA418 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000720	IRQn_IRQVA421 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA420 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000728	IRQn_IRQVA423 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA422 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000730	IRQn_IRQVA425 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA424 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000738	IRQn_IRQVA427 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA426 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000740	IRQn_IRQVA429 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA428 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000748	IRQn_IRQVA431 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA430 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000750	IRQn_IRQVA433 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA432 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000758	IRQn_IRQVA435 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA434 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000760	IRQn_IRQVA437 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA436 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000768	IRQn_IRQVA439 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA438 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000770	IRQn_IRQVA441 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA440 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000778	IRQn_IRQVA443 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA442 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000780	IRQn_IRQVA445 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA444 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000788	IRQn_IRQVA447 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA446 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000790	IRQn_IRQVA449 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA448 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000798	IRQn_IRQVA451 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA450 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007A0	IRQn_IRQVA453 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA452 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007A8	IRQn_IRQVA455 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA454 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000007B0	IRQn_IRQVA457 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA456 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007B8	IRQn_IRQVA459 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA458 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007C0	IRQn_IRQVA461 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA460 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007C8	IRQn_IRQVA463 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA462 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007D0	IRQn_IRQVA465 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA464 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007D8	IRQn_IRQVA467 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA466 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007E0	IRQn_IRQVA469 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA468 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007E8	IRQn_IRQVA471 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA470 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007F0	IRQn_IRQVA473 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA472 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x000007F8	IRQn_IRQVA475 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA474 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000800	IRQn_IRQVA477 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA476 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000808	IRQn_IRQVA479 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA478 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000810	IRQn_IRQVA481 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA480 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000818	IRQn_IRQVA483 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA482 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000820	IRQn_IRQVA485 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA484 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000828	IRQn_IRQVA487 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA486 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000830	IRQn_IRQVA489 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA488 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000838	IRQn_IRQVA491 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA490 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000840	IRQn_IRQVA493 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA492 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000848	IRQn_IRQVA495 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA494 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000850	IRQn_IRQVA497 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA496 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000858	IRQn_IRQVA499 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA498 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000860	IRQn_IRQVA501 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA500 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000868	IRQn_IRQVA503 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA502 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000870	IRQn_IRQVA505 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA504 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000878	IRQn_IRQVA507 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA506 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000880	IRQn_IRQVA509 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA508 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000888	IRQn_IRQVA511 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00				IRQn_IRQVA510 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXX00			
0x00000890	IRQn_NMIPL1 00001111 00001111 00001111 00001111				IRQn_NMIPL0 00001111 00001111 00001111 00000000			
0x00000898	IRQn_NMIPL3 00001111 00001111 00001111 00001111				IRQn_NMIPL2 00001111 00001111 00001111 00001111			
0x000008A0	IRQn_NMIPL5 00001111 00001111 00001111 00001111				IRQn_NMIPL4 00001111 00001111 00001111 00001111			
0x000008A8	IRQn_NMIPL7 00001111 00001111 00001111 00001111				IRQn_NMIPL6 00001111 00001111 00001111 00001111			
0x000008B0	IRQn_IRQPL1 00011111 00011111 00011111 00011111				IRQn_IRQPL0 00011111 00011111 00011111 00011111			
0x000008B8	IRQn_IRQPL3 00011111 00011111 00011111 00011111				IRQn_IRQPL2 00011111 00011111 00011111 00011111			
0x000008C0	IRQn_IRQPL5 00011111 00011111 00011111 00011111				IRQn_IRQPL4 00011111 00011111 00011111 00011111			
0x000008C8	IRQn_IRQPL7 00011111 00011111 00011111 00011111				IRQn_IRQPL6 00011111 00011111 00011111 00011111			
0x000008D0	IRQn_IRQPL9 00011111 00011111 00011111 00011111				IRQn_IRQPL8 00011111 00011111 00011111 00011111			
0x000008D8	IRQn_IRQPL11 00011111 00011111 00011111 00011111				IRQn_IRQPL10 00011111 00011111 00011111 00011111			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000008E0	IRQn_IRQPL13 00011111 00011111 00011111 00011111				IRQn_IRQPL12 00011111 00011111 00011111 00011111			
0x000008E8	IRQn_IRQPL15 00011111 00011111 00011111 00011111				IRQn_IRQPL14 00011111 00011111 00011111 00011111			
0x000008F0	IRQn_IRQPL17 00011111 00011111 00011111 00011111				IRQn_IRQPL16 00011111 00011111 00011111 00011111			
0x000008F8	IRQn_IRQPL19 00011111 00011111 00011111 00011111				IRQn_IRQPL18 00011111 00011111 00011111 00011111			
0x00000900	IRQn_IRQPL21 00011111 00011111 00011111 00011111				IRQn_IRQPL20 00011111 00011111 00011111 00011111			
0x00000908	IRQn_IRQPL23 00011111 00011111 00011111 00011111				IRQn_IRQPL22 00011111 00011111 00011111 00011111			
0x00000910	IRQn_IRQPL25 00011111 00011111 00011111 00011111				IRQn_IRQPL24 00011111 00011111 00011111 00011111			
0x00000918	IRQn_IRQPL27 00011111 00011111 00011111 00011111				IRQn_IRQPL26 00011111 00011111 00011111 00011111			
0x00000920	IRQn_IRQPL29 00011111 00011111 00011111 00011111				IRQn_IRQPL28 00011111 00011111 00011111 00011111			
0x00000928	IRQn_IRQPL31 00011111 00011111 00011111 00011111				IRQn_IRQPL30 00011111 00011111 00011111 00011111			
0x00000930	IRQn_IRQPL33 00011111 00011111 00011111 00011111				IRQn_IRQPL32 00011111 00011111 00011111 00011111			
0x00000938	IRQn_IRQPL35 00011111 00011111 00011111 00011111				IRQn_IRQPL34 00011111 00011111 00011111 00011111			
0x00000940	IRQn_IRQPL37 00011111 00011111 00011111 00011111				IRQn_IRQPL36 00011111 00011111 00011111 00011111			
0x00000948	IRQn_IRQPL39 00011111 00011111 00011111 00011111				IRQn_IRQPL38 00011111 00011111 00011111 00011111			
0x00000950	IRQn_IRQPL41 00011111 00011111 00011111 00011111				IRQn_IRQPL40 00011111 00011111 00011111 00011111			
0x00000958	IRQn_IRQPL43 00011111 00011111 00011111 00011111				IRQn_IRQPL42 00011111 00011111 00011111 00011111			
0x00000960	IRQn_IRQPL45 00011111 00011111 00011111 00011111				IRQn_IRQPL44 00011111 00011111 00011111 00011111			
0x00000968	IRQn_IRQPL47 00011111 00011111 00011111 00011111				IRQn_IRQPL46 00011111 00011111 00011111 00011111			
0x00000970	IRQn_IRQPL49 00011111 00011111 00011111 00011111				IRQn_IRQPL48 00011111 00011111 00011111 00011111			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000978	IRQn_IRQPL51 00011111 00011111 00011111 00011111				IRQn_IRQPL50 00011111 00011111 00011111 00011111			
0x00000980	IRQn_IRQPL53 00011111 00011111 00011111 00011111				IRQn_IRQPL52 00011111 00011111 00011111 00011111			
0x00000988	IRQn_IRQPL55 00011111 00011111 00011111 00011111				IRQn_IRQPL54 00011111 00011111 00011111 00011111			
0x00000990	IRQn_IRQPL57 00011111 00011111 00011111 00011111				IRQn_IRQPL56 00011111 00011111 00011111 00011111			
0x00000998	IRQn_IRQPL59 00011111 00011111 00011111 00011111				IRQn_IRQPL58 00011111 00011111 00011111 00011111			
0x000009A0	IRQn_IRQPL61 00011111 00011111 00011111 00011111				IRQn_IRQPL60 00011111 00011111 00011111 00011111			
0x000009A8	IRQn_IRQPL63 00011111 00011111 00011111 00011111				IRQn_IRQPL62 00011111 00011111 00011111 00011111			
0x000009B0	IRQn_IRQPL65 00011111 00011111 00011111 00011111				IRQn_IRQPL64 00011111 00011111 00011111 00011111			
0x000009B8	IRQn_IRQPL67 00011111 00011111 00011111 00011111				IRQn_IRQPL66 00011111 00011111 00011111 00011111			
0x000009C0	IRQn_IRQPL69 00011111 00011111 00011111 00011111				IRQn_IRQPL68 00011111 00011111 00011111 00011111			
0x000009C8	IRQn_IRQPL71 00011111 00011111 00011111 00011111				IRQn_IRQPL70 00011111 00011111 00011111 00011111			
0x000009D0	IRQn_IRQPL73 00011111 00011111 00011111 00011111				IRQn_IRQPL72 00011111 00011111 00011111 00011111			
0x000009D8	IRQn_IRQPL75 00011111 00011111 00011111 00011111				IRQn_IRQPL74 00011111 00011111 00011111 00011111			
0x000009E0	IRQn_IRQPL77 00011111 00011111 00011111 00011111				IRQn_IRQPL76 00011111 00011111 00011111 00011111			
0x000009E8	IRQn_IRQPL79 00011111 00011111 00011111 00011111				IRQn_IRQPL78 00011111 00011111 00011111 00011111			
0x000009F0	IRQn_IRQPL81 00011111 00011111 00011111 00011111				IRQn_IRQPL80 00011111 00011111 00011111 00011111			
0x000009F8	IRQn_IRQPL83 00011111 00011111 00011111 00011111				IRQn_IRQPL82 00011111 00011111 00011111 00011111			
0x00000A00	IRQn_IRQPL85 00011111 00011111 00011111 00011111				IRQn_IRQPL84 00011111 00011111 00011111 00011111			
0x00000A08	IRQn_IRQPL87 00011111 00011111 00011111 00011111				IRQn_IRQPL86 00011111 00011111 00011111 00011111			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000A10	IRQn_IRQPL89 00011111 00011111 00011111 00011111				IRQn_IRQPL88 00011111 00011111 00011111 00011111			
0x00000A18	IRQn_IRQPL91 00011111 00011111 00011111 00011111				IRQn_IRQPL90 00011111 00011111 00011111 00011111			
0x00000A20	IRQn_IRQPL93 00011111 00011111 00011111 00011111				IRQn_IRQPL92 00011111 00011111 00011111 00011111			
0x00000A28	IRQn_IRQPL95 00011111 00011111 00011111 00011111				IRQn_IRQPL94 00011111 00011111 00011111 00011111			
0x00000A30	IRQn_IRQPL97 00011111 00011111 00011111 00011111				IRQn_IRQPL96 00011111 00011111 00011111 00011111			
0x00000A38	IRQn_IRQPL99 00011111 00011111 00011111 00011111				IRQn_IRQPL98 00011111 00011111 00011111 00011111			
0x00000A40	IRQn_IRQPL101 00011111 00011111 00011111 00011111				IRQn_IRQPL100 00011111 00011111 00011111 00011111			
0x00000A48	IRQn_IRQPL103 00011111 00011111 00011111 00011111				IRQn_IRQPL102 00011111 00011111 00011111 00011111			
0x00000A50	IRQn_IRQPL105 00011111 00011111 00011111 00011111				IRQn_IRQPL104 00011111 00011111 00011111 00011111			
0x00000A58	IRQn_IRQPL107 00011111 00011111 00011111 00011111				IRQn_IRQPL106 00011111 00011111 00011111 00011111			
0x00000A60	IRQn_IRQPL109 00011111 00011111 00011111 00011111				IRQn_IRQPL108 00011111 00011111 00011111 00011111			
0x00000A68	IRQn_IRQPL111 00011111 00011111 00011111 00011111				IRQn_IRQPL110 00011111 00011111 00011111 00011111			
0x00000A70	IRQn_IRQPL113 00011111 00011111 00011111 00011111				IRQn_IRQPL112 00011111 00011111 00011111 00011111			
0x00000A78	IRQn_IRQPL115 00011111 00011111 00011111 00011111				IRQn_IRQPL114 00011111 00011111 00011111 00011111			
0x00000A80	IRQn_IRQPL117 00011111 00011111 00011111 00011111				IRQn_IRQPL116 00011111 00011111 00011111 00011111			
0x00000A88	IRQn_IRQPL119 00011111 00011111 00011111 00011111				IRQn_IRQPL118 00011111 00011111 00011111 00011111			
0x00000A90	IRQn_IRQPL121 00011111 00011111 00011111 00011111				IRQn_IRQPL120 00011111 00011111 00011111 00011111			
0x00000A98	IRQn_IRQPL123 00011111 00011111 00011111 00011111				IRQn_IRQPL122 00011111 00011111 00011111 00011111			
0x00000AA0	IRQn_IRQPL125 00011111 00011111 00011111 00011111				IRQn_IRQPL124 00011111 00011111 00011111 00011111			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000AA8	IRQn_IRQPL127 00011111 00011111 00011111 00011111				IRQn_IRQPL126 00011111 00011111 00011111 00011111			
0x00000AB0	IRQn_NMIR 00000000 00000000 00000000 00000000				IRQn_NMIS 00000000 00000000 00000000 00000000			
0x00000AB8	read0 00000000 00000000 00000000 00000000				IRQn_NMISIS 00000000 00000000 00000000 00000000			
0x00000AC0	IRQn_IRQS1 00000000 00000000 00000000 00000000				IRQn_IRQS0 00000000 00000000 00000000 00000000			
0x00000AC8	IRQn_IRQS3 00000000 00000000 00000000 00000000				IRQn_IRQS2 00000000 00000000 00000000 00000000			
0x00000AD0	IRQn_IRQS5 00000000 00000000 00000000 00000000				IRQn_IRQS4 00000000 00000000 00000000 00000000			
0x00000AD8	IRQn_IRQS7 00000000 00000000 00000000 00000000				IRQn_IRQS6 00000000 00000000 00000000 00000000			
0x00000AE0	IRQn_IRQS9 00000000 00000000 00000000 00000000				IRQn_IRQS8 00000000 00000000 00000000 00000000			
0x00000AE8	IRQn_IRQS11 00000000 00000000 00000000 00000000				IRQn_IRQS10 00000000 00000000 00000000 00000000			
0x00000AF0	IRQn_IRQS13 00000000 00000000 00000000 00000000				IRQn_IRQS12 00000000 00000000 00000000 00000000			
0x00000AF8	IRQn_IRQS15 00000000 00000000 00000000 00000000				IRQn_IRQS14 00000000 00000000 00000000 00000000			
0x00000B00	IRQn_IRQR1 00000000 00000000 00000000 00000000				IRQn_IRQR0 00000000 00000000 00000000 00000000			
0x00000B08	IRQn_IRQR3 00000000 00000000 00000000 00000000				IRQn_IRQR2 00000000 00000000 00000000 00000000			
0x00000B10	IRQn_IRQR5 00000000 00000000 00000000 00000000				IRQn_IRQR4 00000000 00000000 00000000 00000000			
0x00000B18	IRQn_IRQR7 00000000 00000000 00000000 00000000				IRQn_IRQR6 00000000 00000000 00000000 00000000			
0x00000B20	IRQn_IRQR9 00000000 00000000 00000000 00000000				IRQn_IRQR8 00000000 00000000 00000000 00000000			
0x00000B28	IRQn_IRQR11 00000000 00000000 00000000 00000000				IRQn_IRQR10 00000000 00000000 00000000 00000000			
0x00000B30	IRQn_IRQR13 00000000 00000000 00000000 00000000				IRQn_IRQR12 00000000 00000000 00000000 00000000			
0x00000B38	IRQn_IRQR15 00000000 00000000 00000000 00000000				IRQn_IRQR14 00000000 00000000 00000000 00000000			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000B40	IRQn_IRQSIS1 00000000 00000000 00000000 00000000				IRQn_IRQSIS0 00000000 00000000 00000000 00000000			
0x00000B48	IRQn_IRQSIS3 00000000 00000000 00000000 00000000				IRQn_IRQSIS2 00000000 00000000 00000000 00000000			
0x00000B50	IRQn_IRQSIS5 00000000 00000000 00000000 00000000				IRQn_IRQSIS4 00000000 00000000 00000000 00000000			
0x00000B58	IRQn_IRQSIS7 00000000 00000000 00000000 00000000				IRQn_IRQSIS6 00000000 00000000 00000000 00000000			
0x00000B60	IRQn_IRQSIS9 00000000 00000000 00000000 00000000				IRQn_IRQSIS8 00000000 00000000 00000000 00000000			
0x00000B68	IRQn_IRQSIS11 00000000 00000000 00000000 00000000				IRQn_IRQSIS10 00000000 00000000 00000000 00000000			
0x00000B70	IRQn_IRQSIS13 00000000 00000000 00000000 00000000				IRQn_IRQSIS12 00000000 00000000 00000000 00000000			
0x00000B78	IRQn_IRQSIS15 00000000 00000000 00000000 00000000				IRQn_IRQSIS14 00000000 00000000 00000000 00000000			
0x00000B80	IRQn_IRQCES1 00000000 00000000 00000000 00000000				IRQn_IRQCES0 00000000 00000000 00000000 00000000			
0x00000B88	IRQn_IRQCES3 00000000 00000000 00000000 00000000				IRQn_IRQCES2 00000000 00000000 00000000 00000000			
0x00000B90	IRQn_IRQCES5 00000000 00000000 00000000 00000000				IRQn_IRQCES4 00000000 00000000 00000000 00000000			
0x00000B98	IRQn_IRQCES7 00000000 00000000 00000000 00000000				IRQn_IRQCES6 00000000 00000000 00000000 00000000			
0x00000BA0	IRQn_IRQCES9 00000000 00000000 00000000 00000000				IRQn_IRQCES8 00000000 00000000 00000000 00000000			
0x00000BA8	IRQn_IRQCES11 00000000 00000000 00000000 00000000				IRQn_IRQCES10 00000000 00000000 00000000 00000000			
0x00000BB0	IRQn_IRQCES13 00000000 00000000 00000000 00000000				IRQn_IRQCES12 00000000 00000000 00000000 00000000			
0x00000BB8	IRQn_IRQCES15 00000000 00000000 00000000 00000000				IRQn_IRQCES14 00000000 00000000 00000000 00000000			
0x00000BC0	IRQn_IRQCEC1 00000000 00000000 00000000 00000000				IRQn_IRQCEC0 00000000 00000000 00000000 00000000			
0x00000BC8	IRQn_IRQCEC3 00000000 00000000 00000000 00000000				IRQn_IRQCEC2 00000000 00000000 00000000 00000000			
0x00000BD0	IRQn_IRQCEC5 00000000 00000000 00000000 00000000				IRQn_IRQCEC4 00000000 00000000 00000000 00000000			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000BD8	IRQn_IRQCEC7 00000000 00000000 00000000 00000000				IRQn_IRQCEC6 00000000 00000000 00000000 00000000			
0x00000BE0	IRQn_IRQCEC9 00000000 00000000 00000000 00000000				IRQn_IRQCEC8 00000000 00000000 00000000 00000000			
0x00000BE8	IRQn_IRQCEC11 00000000 00000000 00000000 00000000				IRQn_IRQCEC10 00000000 00000000 00000000 00000000			
0x00000BF0	IRQn_IRQCEC13 00000000 00000000 00000000 00000000				IRQn_IRQCEC12 00000000 00000000 00000000 00000000			
0x00000BF8	IRQn_IRQCEC15 00000000 00000000 00000000 00000000				IRQn_IRQCEC14 00000000 00000000 00000000 00000000			
0x00000C00	IRQn_IRQCE1 00000000 00000000 00000000 00000000				IRQn_IRQCE0 00000000 00000000 00000000 00000000			
0x00000C08	IRQn_IRQCE3 00000000 00000000 00000000 00000000				IRQn_IRQCE2 00000000 00000000 00000000 00000000			
0x00000C10	IRQn_IRQCE5 00000000 00000000 00000000 00000000				IRQn_IRQCE4 00000000 00000000 00000000 00000000			
0x00000C18	IRQn_IRQCE7 00000000 00000000 00000000 00000000				IRQn_IRQCE6 00000000 00000000 00000000 00000000			
0x00000C20	IRQn_IRQCE9 00000000 00000000 00000000 00000000				IRQn_IRQCE8 00000000 00000000 00000000 00000000			
0x00000C28	IRQn_IRQCE11 00000000 00000000 00000000 00000000				IRQn_IRQCE10 00000000 00000000 00000000 00000000			
0x00000C30	IRQn_IRQCE13 00000000 00000000 00000000 00000000				IRQn_IRQCE12 00000000 00000000 00000000 00000000			
0x00000C38	IRQn_IRQCE15 00000000 00000000 00000000 00000000				IRQn_IRQCE14 00000000 00000000 00000000 00000000			
0x00000C40	IRQn_NMIHS 00000000 00000000 00000000 00000000				IRQn_NMIHC 00000000 00000000 00000000 00000000			
0x00000C48	read0 00000000 00000000 00000000 00000000				IRQn_IRQHC 00000000 00000000 00000000 00000000			
0x00000C50	IRQn_IRQHS1 00000000 00000000 00000000 00000000				IRQn_IRQHS0 00000000 00000000 00000000 00000000			
0x00000C58	IRQn_IRQHS3 00000000 00000000 00000000 00000000				IRQn_IRQHS2 00000000 00000000 00000000 00000000			
0x00000C60	IRQn_IRQHS5 00000000 00000000 00000000 00000000				IRQn_IRQHS4 00000000 00000000 00000000 00000000			
0x00000C68	IRQn_IRQHS7 00000000 00000000 00000000 00000000				IRQn_IRQHS6 00000000 00000000 00000000 00000000			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000C70	IRQn_IRQHS9 00000000 00000000 00000000 00000000				IRQn_IRQHS8 00000000 00000000 00000000 00000000			
0x00000C78	IRQn_IRQHS11 00000000 00000000 00000000 00000000				IRQn_IRQHS10 00000000 00000000 00000000 00000000			
0x00000C80	IRQn_IRQHS13 00000000 00000000 00000000 00000000				IRQn_IRQHS12 00000000 00000000 00000000 00000000			
0x00000C88	IRQn_IRQHS15 00000000 00000000 00000000 00000000				IRQn_IRQHS14 00000000 00000000 00000000 00000000			
0x00000C90	read0 00000000 00000000 00000000 00000000				IRQn_IRQPLM 00000000 00000000 00000000 00100000			
0x00000C98	read0 00000000 00000000 00000000 00000000				IRQn_CSR 00000000 00000001 00000000 00000000			
0x00000CA0	read0 00000000 00000000 00000000 00000000				IRQn_NESTL 00000000 00000000 00000000 00000000			
0x00000CA8	IRQn_NMIPS 00000000 00000000 00000000 00000000				IRQn_NMIRS 00000000 00000000 00000000 00000000			
0x00000CB0	IRQn_IRQRS1 00000000 00000000 00000000 00000000				IRQn_IRQRS0 00000000 00000000 00000000 00000000			
0x00000CB8	IRQn_IRQRS3 00000000 00000000 00000000 00000000				IRQn_IRQRS2 00000000 00000000 00000000 00000000			
0x00000CC0	IRQn_IRQRS5 00000000 00000000 00000000 00000000				IRQn_IRQRS4 00000000 00000000 00000000 00000000			
0x00000CC8	IRQn_IRQRS7 00000000 00000000 00000000 00000000				IRQn_IRQRS6 00000000 00000000 00000000 00000000			
0x00000CD0	IRQn_IRQRS9 00000000 00000000 00000000 00000000				IRQn_IRQRS8 00000000 00000000 00000000 00000000			
0x00000CD8	IRQn_IRQRS11 00000000 00000000 00000000 00000000				IRQn_IRQRS10 00000000 00000000 00000000 00000000			
0x00000CE0	IRQn_IRQRS13 00000000 00000000 00000000 00000000				IRQn_IRQRS12 00000000 00000000 00000000 00000000			
0x00000CE8	IRQn_IRQRS15 00000000 00000000 00000000 00000000				IRQn_IRQRS14 00000000 00000000 00000000 00000000			
0x00000CF0	IRQn_IRQPS1 00000000 00000000 00000000 00000000				IRQn_IRQPS0 00000000 00000000 00000000 00000000			
0x00000CF8	IRQn_IRQPS3 00000000 00000000 00000000 00000000				IRQn_IRQPS2 00000000 00000000 00000000 00000000			
0x00000D00	IRQn_IRQPS5 00000000 00000000 00000000 00000000				IRQn_IRQPS4 00000000 00000000 00000000 00000000			

Table 17-1. Memory layout of IUNIT register

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000D08	IRQn_IRQPS7 00000000 00000000 00000000 00000000				IRQn_IRQPS6 00000000 00000000 00000000 00000000			
0x00000D10	IRQn_IRQPS9 00000000 00000000 00000000 00000000				IRQn_IRQPS8 00000000 00000000 00000000 00000000			
0x00000D18	IRQn_IRQPS11 00000000 00000000 00000000 00000000				IRQn_IRQPS10 00000000 00000000 00000000 00000000			
0x00000D20	IRQn_IRQPS13 00000000 00000000 00000000 00000000				IRQn_IRQPS12 00000000 00000000 00000000 00000000			
0x00000D28	IRQn_IRQPS15 00000000 00000000 00000000 00000000				IRQn_IRQPS14 00000000 00000000 00000000 00000000			
0x00000D30	read0 00000000 00000000 00000000 00000000				IRQn_UNLOCK 00000000 00000000 00000000 00000000			
0x00000D38	read0 00000000 00000000 00000000 00000000				IRQn_MID 00000000 00000000 00000000 00000000			
0x00000D40	IRQn_EAN 00000000 00000000 00000000 00000000				IRQn_EEI 00000000 00000000 00000000 00000000			
0x00000D48	IRQn_EEB0 00000000 00000000 00000000 00000000				IRQn_ET 00000000 00000000 00000000 00000000			
0x00000D50	IRQn_EEB2 00000000 00000000 00000000 00000000				IRQn_EEB1 00000000 00000000 00000000 00000000			

Notes:

1. Each instance of a IUNIT module uses 4 KBs of CPU's address space for mapping its own Control and Status Registers (CSRs). The registers can be accessed with 8-/16-/32-/64-bit accesses.
2. For AHB read accesses, the behaviour of bytes defined as read0 in certain registers (IRQn_NMIST[31:16], IRQn_IRQST[31:24], IRQn_NMIHC[31:8], IRQn_IRQHC[31:16], IRQn_IRQPLM[31:8], IRQn_CSR[31:24] and IRQn_CSR[15:8], IRQn_NESTL[31:16], IRQn_EAN[31:8], IRQn_ET[31:8] and IRQn_EEB2[31:16]) behave as reserved. This is dependent on the following condition:
   - If an interrupt channel is not enabled, the behavior of the register (i.e. as either read0 or reserved) depends on whether or not neighbouring channels are enabled. In other words, if a channel has a register associated with it on a 64-bit address width, a register not enabled on the same 64-bit address width will behave as either read0 or reserved. For most registers, this relationship is true between two neighbouring channels. For registers which store the priority bits NMIPL and IRQPL, this relationship is extended to 8 neighbouring channels.
3. Writing to the IRQn_NMIS and IRQ_NMIR registers at the same time is allowed as long as the two registers do not operate on the same channel.
4. Reading the IRQn_NMIVAS and IRQn_NMIST registers at the same time in user mode will return an error response.

17.2.1 IUNIT NMI Vector Address Status Register (IRQn_NMIVAS)

This register provides vector address status of NMI interrupt last applied to the CPU. CPU can read this register to branch to the ISR of NMI.

IUNIT NMI Vector Address Status Register (IRQn_NMIVAS)

Figure 17-2. IUNIT NMI Vector Address Status Register (IRQn_NMIVAS)

IRQn_NMIVAS																															
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**
NMIVAS[31]	NMIVAS[30]	NMIVAS[29]	NMIVAS[28]	NMIVAS[27]	NMIVAS[26]	NMIVAS[25]	NMIVAS[24]	NMIVAS[23]	NMIVAS[22]	NMIVAS[21]	NMIVAS[20]	NMIVAS[19]	NMIVAS[18]	NMIVAS[17]	NMIVAS[16]	NMIVAS[15]	NMIVAS[14]	NMIVAS[13]	NMIVAS[12]	NMIVAS[11]	NMIVAS[10]	NMIVAS[9]	NMIVAS[8]	NMIVAS[7]	NMIVAS[6]	NMIVAS[5]	NMIVAS[4]	NMIVAS[3]	NMIVAS[2]	NMIVAS[1]	NMIVAS[0]
RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP	RP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-2. IUNIT NMI Vector Address Status Register (IRQn_NMIVAS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMIVAS	NMI Vector Address Status Vector address of NMI last applied to the CPU.

Note: A copy of IRQn_NMIVAS register is maintained at an address location 0x3FC in a separate 1 KB address space.

17.2.2 IUNIT NMI Status Register (IRQn_NMIST)

This register provides priority level and NMI source number of NMI interrupt last applied to the CPU.

IUNIT NMI Status Register (IRQn_NMIST)

Figure 17-3. IUNIT NMI Status Register (IRQn_NMIST)

IRQn_NMIST																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMIPS[3]	NMIPS[2]	NMIPS[1]	NMIPS[0]	read0	read0	NMISN[5]	NMISN[4]	NMISN[3]	NMISN[2]	NMISN[1]	NMISN[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R0	R0	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0

Table 17-3. IUNIT NMI Status Register (IRQn_NMIST) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:12]	read0	-
[11:8]	NMIPS	NMI Priority Status Priority level of last NMI applied to the CPU.
[7:6]	read0	-
[5:0]	NMISN	NMI Source Number NMI source number of NMI last applied to the CPU. 0x0: NMI source 0 0x1: NMI source 1 0x2: NMI source 2 ... 0x1F:NMI source 31

17.2.3 IUNIT IRQ Vector Address Status Register (IRQn_IRQVAS)

This register provides vector address status of IRQ interrupt last applied to the CPU.

IUNIT IRQ Vector Address Status Register (IRQn_IRQVAS)

Figure 17-4. IUNIT IRQ Vector Address Status Register (IRQn_IRQVAS)

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

Table 17-4. IUNIT IRQ Vector Address Status Register (IRQn_IRQVAS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQVAS	IRQ Vector Address Status Vector address of IRQ last applied to the CPU.

17.2.4 IUNIT IRQ Status Register (IRQn_IRQST)

This register provides priority level and IRQ source number of IRQ interrupt last applied to the CPU.

IUNIT IRQ Status Register (IRQn_IRQST)

Figure 17-5. IUNIT IRQ Status Register (IRQn_IRQST)

IRQn_IRQST																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IRQPS[4]	IRQPS[3]	IRQPS[2]	IRQPS[1]	IRQPS[0]	read0	read0	read0	read0	read0	read0	IRQSN[9]	IRQSN[8]	IRQSN[7]	IRQSN[6]	IRQSN[5]	IRQSN[4]	IRQSN[3]	IRQSN[2]	IRQSN[1]	IRQSN[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 17-5. IUNIT IRQ Status Register (IRQn_IRQST) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:21]	read0	-
[20:16]	IRQPS	IRQ Priority Status Priority level of last IRQ applied to the CPU.
[15:10]	read0	-
[9:0]	IRQSN	IRQ Source Number IRQ source of IRQ last applied to the CPU. 0x0: IRQ source 0 0x1: IRQ source 1 0x2: IRQ source 2 ... 0x1FF:IRQ source 511

17.2.5 IUNIT NMI Vector Address Register (IRQn_NMIVA0~31)

These registers provide 32-bit vector address of 32 NMI interrupt source. Software must initialize the vector address before enabling the corresponding NMI peripheral.

IUNIT NMI Vector Address Register (IRQn_NMIVA0)

Figure 17-6. IUNIT NMI Vector Address Register (IRQn_NMIVA0)

IRQn_NMIVA0																															
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**
NMIVA[31]	NMIVA[30]	NMIVA[29]	NMIVA[28]	NMIVA[27]	NMIVA[26]	NMIVA[25]	NMIVA[24]	NMIVA[23]	NMIVA[22]	NMIVA[21]	NMIVA[20]	NMIVA[19]	NMIVA[18]	NMIVA[17]	NMIVA[16]	NMIVA[15]	NMIVA[14]	NMIVA[13]	NMIVA[12]	NMIVA[11]	NMIVA[10]	NMIVA[9]	NMIVA[8]	NMIVA[7]	NMIVA[6]	NMIVA[5]	NMIVA[4]	NMIVA[3]	NMIVA[2]	read0	read0
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	R0	R0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Table 17-6. IUNIT NMI Vector Address Register (IRQn_NMIVA0) bits

Bit Position	Bit Field Name	Bit Description
[31:2]	NMIVA	Vector Address of NMI ISR vector address of NMI0. The first default address value (for IRQ0_NMIVA0) is 0xFFFF0040. The default vector address value for each of the following 31 registers is then continuously incremented by 4. Moreover, these default vector addresses all cause a software triggered hardware reset.
[1:0]	read0	-

Notes:

1. All vector addresses are 32-bit aligned, hence the value on the lower 2 bits of this register are ignored by the IUNIT. On reading this register the lower 2 bits will always be read zero.
2. IRQn_NMIVA0 provides ISR vector address for NMI source 0 (NMI0). This register is repeated for each NMI source. Since IUNIT supports 32 NMI sources, there exist 32 such registers (IRQn_NMIVA0 to IRQn_NMIVA31) corresponding to each NMI sources (NMI0 to NMI31).

17.2.6 IUNIT IRQ Vector Address Register (IRQn_IRQVA0~511)

These registers provide 32-bit vector address of IRQ interrupt source. Software must initialize the vector address before enabling the corresponding interrupt channel.

IUNIT IRQ Vector Address Register (IRQn_IRQVA0)

Figure 17-7. IUNIT IRQ Vector Address Register (IRQn_IRQVA0)

IRQn_IRQVA0																															
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**
IRQVA[31]	IRQVA[30]	IRQVA[29]	IRQVA[28]	IRQVA[27]	IRQVA[26]	IRQVA[25]	IRQVA[24]	IRQVA[23]	IRQVA[22]	IRQVA[21]	IRQVA[20]	IRQVA[19]	IRQVA[18]	IRQVA[17]	IRQVA[16]	IRQVA[15]	IRQVA[14]	IRQVA[13]	IRQVA[12]	IRQVA[11]	IRQVA[10]	IRQVA[9]	IRQVA[8]	IRQVA[7]	IRQVA[6]	IRQVA[5]	IRQVA[4]	IRQVA[3]	IRQVA[2]	read0	read0
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	R0	R0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Table 17-7. IUNIT IRQ Vector Address Register (IRQn_IRQVA0) bits

Bit Position	Bit Field Name	Bit Description
[31:2]	IRQVA	Vector Address of IRQ ISR vector address of IRQ0.
[1:0]	read0	-

Notes:

1. The software must only do an 8-bit, 16-bit, 32-bit, or 64-bit read access on these registers, after address initialization. The address initialization should be done only by 32-bit or 64-bit access.
2. All vector addresses are 32-bit aligned, hence the value on the lower 2 bits of this register are ignored by the IUNIT. On reading this register the lower 2 bits will always be read zeroes.
3. IRQn_IRQVA0 provides ISR vector address for IRQ source 0 (IRQ0). This register is repeated for each IRQ source. Since IUNIT supports 512 IRQ sources, there exist 512 such registers (IRQn_IRQVA0 to IRQn_IRQVA511) corresponding to each IRQ sources (IRQ0 to IRQ511).

17.2.7 IUNIT NMI Priority Level Register (IRQn_NMIPL0)

This register provides priority levels corresponding to NMI sources NMI0 to NMI3. The NMIPL can be configured to any value between 0 to 15. The lower the value of the individual NMIPL, higher the priority of that NMI. By default all the NMIPL are configured to their lowest priority (i.e. 15). As a special case, the priority of NMI source 0 is hard wired to 0.

IUNIT NMI Priority Level Register (IRQn_NMIPL0)

Figure 17-8. IUNIT NMI Priority Level Register (IRQn_NMIPL0)

IRQn_NMIPL1																															
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**
read0	read0	read0	read0	NMIPL3[3]	NMIPL3[2]	NMIPL3[1]	NMIPL3[0]	read0	read0	read0	read0	NMIPL2[3]	NMIPL2[2]	NMIPL2[1]	NMIPL2[0]	read0	read0	read0	read0	NMIPL1[3]	NMIPL1[2]	NMIPL1[1]	NMIPL1[0]	read0	read0	read0	read0	NMIPL0[3]	NMIPL0[2]	NMIPL0[1]	NMIPL0[0]
R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Table 17-8. IUNIT NMI Priority Level Register (IRQn_NMIPL0) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:24]	NMIPL3	NMI Priority Level Priority level for NMI3.
[23:20]	read0	-
[19:16]	NMIPL2	NMI Priority Level Priority level for NMI2.
[15:12]	read0	-
[11:8]	NMIPL1	NMI Priority Level Priority level for NMI1.
[7:4]	read0	-
[3:0]	NMIPL0	NMI Priority Level Priority level for NMI0, hardwired to '0'.

17.2.8 IUNIT NMI Priority Level Register (IRQn_NMIPL1~7)

These registers provide priority levels corresponding to NMI sources NMI4 to NMI31. The IRQn_NMIPL:NMIPL can be configured to any value between 0 to 15. Lower the value of the individual IRQn_NMIPL:NMIPL, higher is the priority of that NMI. By default all the IRQn_NMIPL:NMIPL are configured to their lowest priority (i.e. 15).

IUNIT NMI Priority Level Register (IRQn_NMIPL1)

Figure 17-9. IUNIT NMI Priority Level Register (IRQn_NMIPL1)

IRQn_NMIPL1																													
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**
read0	read0	read0	read0	NMIPL3[3]	NMIPL3[2]	NMIPL3[1]	NMIPL3[0]	read0	read0	read0	read0	NMIPL2[3]	NMIPL2[2]	NMIPL2[1]	NMIPL2[0]	read0	read0	read0	read0	NMIPL1[3]	NMIPL1[2]	NMIPL1[1]	NMIPL1[0]	read0	read0	read0	read0	NMIPL0[3]	NMIPL0[2]	NMIPL0[1]	NMIPL0[0]
R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	RWP	RWP	RWP	RWP
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

Table 17-9. IUNIT NMI Priority Level Register (IRQn_NMIPL1) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:24]	NMIPL3	NMI Priority Level Priority level for NMI7.
[23:20]	read0	-
[19:16]	NMIPL2	NMI Priority Level Priority level for NMI6.
[15:12]	read0	-
[11:8]	NMIPL1	NMI Priority Level Priority level for NMI5.
[7:4]	read0	-
[3:0]	NMIPL0	NMI Priority Level Priority level for NMI4.

Note: IRQn_NMIPL1 provides priority levels for NMI source 4 to 7 (NMI4 to NMI7), IRQn_NMIPL2 provides priority levels for NMI source 8 to 11 (NMI8 to NMI11). The last register IRQn_NMIPL7 provides priority levels for NMI source 28 to 31 (NMI28 to NMI31).

17.2.9 IUNIT IRQ Priority Level Register (IRQn_IRQPL0~127)

These registers provide priority levels corresponding to each IRQ sources. The IRQn_IRQPL:IRQPL bits can be programmed to any value between 0 to 31. The lower the value of the individual IRQn_IRQPL:IRQPL, higher the priority of the corresponding IRQ. By default all the IRQn_IRQPL:IRQPL are programmed to their lowest priority (i.e. 31).

IUNIT IRQ Priority Level Register (IRQn_IRQPL0)

Figure 17-10. IUNIT IRQ Priority Level Register (IRQn_IRQPL0)

IRQn_IRQPL0																															
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**
read0	read0	read0	IRQPL3[4]	IRQPL3[3]	IRQPL3[2]	IRQPL3[1]	IRQPL3[0]	read0	read0	read0	IRQPL2[4]	IRQPL2[3]	IRQPL2[2]	IRQPL2[1]	IRQPL2[0]	read0	read0	read0	IRQPL1[4]	IRQPL1[3]	IRQPL1[2]	IRQPL1[1]	IRQPL1[0]	read0	read0	read0	IRQPL0[4]	IRQPL0[3]	IRQPL0[2]	IRQPL0[1]	IRQPL0[0]
R0	R0	R0	RWP	RWP	RWP	RWP	RWP	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	R0	R0	R0	RWP	RWP	RWP	RWP	RWP
0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1

Table 17-10. IUNIT IRQ Priority Level Register (IRQn_IRQPL0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:24]	IRQPL3	Normal Interrupt Priority Level Priority level for IRQ3.
[23:21]	read0	-
[20:16]	IRQPL2	Normal Interrupt Priority Level Priority level for IRQ2.
[15:13]	read0	-
[12:8]	IRQPL1	Normal Interrupt Priority Level Priority level for IRQ1.
[7:5]	read0	-
[4:0]	IRQPL0	Normal Interrupt Priority Level Priority level for IRQ0.

Note: IRQn_IRQPL0 provides priority levels for IRQ source 0 to 3 (IRQ0 to IRQ3). As one register accommodates priority levels for 4 sources, there exist 128 such registers (IRQn_IRQPL0 to IRQn_IRQPL127) to cover all IRQ sources (IRQ0 to IRQ511). IRQn_IRQPL1 provides priority levels for IRQ source 4 to 7 (IRQ4 to IRQ7), IRQn_IRQPL2 provides priority levels for IRQ source 8 to 11 (IRQ8 to IRQ11). The last register IRQn_IRQPL127 provides priority levels for IRQ source 508 to 511 (IRQ508 to IRQ511).

17.2.10 IUNIT NMI Set Register (IRQn_NMIS)

Software can set the individual NMI software interrupts using this register.

IUNIT NMI Set Register (IRQn_NMIS)

Figure 17-11. IUNIT NMI Set Register (IRQn_NMIS)

IRQn_NMIS																															
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**
NMIS[31]	NMIS[30]	NMIS[29]	NMIS[28]	NMIS[27]	NMIS[26]	NMIS[25]	NMIS[24]	NMIS[23]	NMIS[22]	NMIS[21]	NMIS[20]	NMIS[19]	NMIS[18]	NMIS[17]	NMIS[16]	NMIS[15]	NMIS[14]	NMIS[13]	NMIS[12]	NMIS[11]	NMIS[10]	NMIS[9]	NMIS[8]	NMIS[7]	NMIS[6]	NMIS[5]	NMIS[4]	NMIS[3]	NMIS[2]	NMIS[1]	NMIS[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-11. IUNIT NMI Set Register (IRQn_NMIS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMIS	Software NMI Set '0': No effect '1': Sets the corresponding NMI software interrupt This register always reads '0'.

Note: NMI software interrupt NMI31 to NMI0 can be set by this register IRQn_NMIS.

17.2.11 IUNIT NMI Reset Register (IRQn_NMIR)

Software can reset the individual NMI software interrupts using this register.

IUNIT NMI Reset Register (IRQn_NMIR)

Figure 17-12. IUNIT NMI Reset Register (IRQn_NMIR)

IRQn_NMIR																															
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**
NMIR[31]	NMIR[30]	NMIR[29]	NMIR[28]	NMIR[27]	NMIR[26]	NMIR[25]	NMIR[24]	NMIR[23]	NMIR[22]	NMIR[21]	NMIR[20]	NMIR[19]	NMIR[18]	NMIR[17]	NMIR[16]	NMIR[15]	NMIR[14]	NMIR[13]	NMIR[12]	NMIR[11]	NMIR[10]	NMIR[9]	NMIR[8]	NMIR[7]	NMIR[6]	NMIR[5]	NMIR[4]	NMIR[3]	NMIR[2]	NMIR[1]	NMIR[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-12. IUNIT NMI Reset Register (IRQn_NMIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMIR	Software NMI Reset '0': No effect '1': Resets the corresponding NMI software interrupt This register always reads '0'.

Note: NMI software interrupt NMI31 to NMI0 can be reset by this register IRQn_NMIS.

17.2.12 IUNIT NMI Software Interrupt Status Register (IRQn_NMISIS)

This register provides the NMI software interrupt status.

IUNIT NMI Software Status Register (IRQn_NMISIS)

Figure 17-13. IUNIT NMI Software Status Register (IRQn_NMISIS)

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

Table 17-13. IUNIT NMI Software Status Register (IRQn_NMISIS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMISIS	NMI Software Interrupt Status Each bit provides the status of corresponding NMI software interrupt. '0': Software interrupt is not set or cleared '1': Software interrupt is set

Note: IRQn_NMISIS provides NMI software interrupt status for NMI31 to NMI0.

17.2.13 IUNIT IRQ Set Register (IRQn_IRQS0~15)

Software can set the individual IRQ software interrupts using these registers.

IUNIT IRQ Set Register (IRQn_IRQS0)

Figure 17-14. IUNIT IRQ Set Register (IRQn_IRQS0)

IRQn_IRQS0																															
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**
IRQS[31]	IRQS[30]	IRQS[29]	IRQS[28]	IRQS[27]	IRQS[26]	IRQS[25]	IRQS[24]	IRQS[23]	IRQS[22]	IRQS[21]	IRQS[20]	IRQS[19]	IRQS[18]	IRQS[17]	IRQS[16]	IRQS[15]	IRQS[14]	IRQS[13]	IRQS[12]	IRQS[11]	IRQS[10]	IRQS[9]	IRQS[8]	IRQS[7]	IRQS[6]	IRQS[5]	IRQS[4]	IRQS[3]	IRQS[2]	IRQS[1]	IRQS[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-14. IUNIT IRQ Set Register (IRQn_IRQS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQS	Software IRQ Set '0': No effect '1': Sets the corresponding IRQ software interrupt This register always reads '0'. The bits in this register correspond to IRQ[31:0].

Note: IRQn_IRQS0 provides set bits for software IRQ interrupt 0 to 31 (IRQ0 to IRQ31). As one register accommodates software set interrupt for 32 sources, there exist 16 such registers (IRQn_IRQS0 to IRQn_IRQS15) to cover all IRQ sources (IRQ0 to IRQ511). IRQn_IRQS1 provides software set interrupt for IRQ source 32 to 63 (IRQ32 to IRQ63), IRQn_IRQS2 provides software set interrupt for IRQ source 64 to 96 (IRQ64 to IRQ95). The last register IRQn_IRQS15 provides software set interrupt for IRQ source 480 to 511 (IRQ480 to IRQ511).

17.2.14 IUNIT IRQ Reset Register (IRQn_IRQR0~15)

Software can reset the individual IRQ software interrupts using these registers.

IUNIT Reset IRQ Register (IRQn_IRQR0)

Figure 17-15. IUNIT IRQ Reset Register (IRQn_IRQR0)

IRQn_IRQR0																															
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**
IRQR[31]	IRQR[30]	IRQR[29]	IRQR[28]	IRQR[27]	IRQR[26]	IRQR[25]	IRQR[24]	IRQR[23]	IRQR[22]	IRQR[21]	IRQR[20]	IRQR[19]	IRQR[18]	IRQR[17]	IRQR[16]	IRQR[15]	IRQR[14]	IRQR[13]	IRQR[12]	IRQR[11]	IRQR[10]	IRQR[9]	IRQR[8]	IRQR[7]	IRQR[6]	IRQR[5]	IRQR[4]	IRQR[3]	IRQR[2]	IRQR[1]	IRQR[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-15. IUNIT IRQ Reset Register (IRQn_IRQR0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQR	Software IRQ Reset '0': No effect '1': Resets the corresponding IRQ software interrupt This register always reads '0'. The bits in this register correspond to IRQ[31:0].

Note: IRQn_IRQR0 provides reset bits for software IRQ interrupt 0 to 31 (IRQ0 to IRQ31). As one register accommodates software reset interrupt for 32 sources, there exist 16 such registers (IRQn_IRQR0 to IRQn_IRQR15) to cover all IRQ sources (IRQ0 to IRQ511). IRQn_IRQR1 provides software reset interrupt for IRQ source 32 to 63 (IRQ32 to IRQ63), IRQn_IRQR2 provides software reset interrupt for IRQ source 64 to 96 (IRQ64 to IRQ95). The last register IRQn_IRQR15 provides software reset interrupt for IRQ source 480 to 511 (IRQ480 to IRQ511).

17.2.15 IUNIT IRQ Software Interrupt Status Register (IRQn_IRQSIS0~15)

These registers provide the IRQ software interrupt status.

IUNIT IRQ Software Interrupt Status Register (IRQn_IRQSIS0)

Figure 17-16. IUNIT IRQ Software Interrupt Status Register (IRQn_IRQSIS0)

IRQn_IRQSIS0																															
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**
IRQSIS[31]	IRQSIS[30]	IRQSIS[29]	IRQSIS[28]	IRQSIS[27]	IRQSIS[26]	IRQSIS[25]	IRQSIS[24]	IRQSIS[23]	IRQSIS[22]	IRQSIS[21]	IRQSIS[20]	IRQSIS[19]	IRQSIS[18]	IRQSIS[17]	IRQSIS[16]	IRQSIS[15]	IRQSIS[14]	IRQSIS[13]	IRQSIS[12]	IRQSIS[11]	IRQSIS[10]	IRQSIS[9]	IRQSIS[8]	IRQSIS[7]	IRQSIS[6]	IRQSIS[5]	IRQSIS[4]	IRQSIS[3]	IRQSIS[2]	IRQSIS[1]	IRQSIS[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-16. IUNIT IRQ Software Interrupt Status Register (IRQn_IRQSIS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQSIS	IRQ Software Interrupt Status Each bit provides the status of corresponding IRQ software interrupt. '0': Software Interrupt is not set or cleared '1': Software Interrupt is set The bits in this register correspond to IRQ software interrupt IRQ[31:0].

Note: IRQn_IRQSIS0 provides status of software IRQ interrupt 0 to 31 (IRQ0 to IRQ31). As one register accommodates software interrupt status for 32 sources, there exist 16 such registers (IRQn_IRQSIS0 to IRQn_IRQSIS15) to cover all software IRQ sources (IRQ0 to IRQ511). IRQn_IRQSIS1 provides status of software IRQ interrupts 32 to 63 (IRQ32 to IRQ63), IRQ_IRQSIS2 provides status of software IRQ interrupts 64 to 96 (IRQ64 to IRQ95). The last register IRQn_IRQSIS15 provides status of software IRQ interrupts 480 to 511 (IRQ480 to IRQ511).

17.2.16 IUNIT IRQ Channel Enable Set Register (IRQn_IRQCES0~15)

The software can use these registers to set the individual IRQ channels enables in IRQn_IRQCE0~15n registers.

IUNIT IRQ Channel Enable Set Register (IRQn_IRQCES0)

Figure 17-17. IUNIT IRQ Channel Enable Set Register (IRQn_IRQCES0) bits

IRQn_IRQCES0																															
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**
IRQCES[31]	IRQCES[30]	IRQCES[29]	IRQCES[28]	IRQCES[27]	IRQCES[26]	IRQCES[25]	IRQCES[24]	IRQCES[23]	IRQCES[22]	IRQCES[21]	IRQCES[20]	IRQCES[19]	IRQCES[18]	IRQCES[17]	IRQCES[16]	IRQCES[15]	IRQCES[14]	IRQCES[13]	IRQCES[12]	IRQCES[11]	IRQCES[10]	IRQCES[9]	IRQCES[8]	IRQCES[7]	IRQCES[6]	IRQCES[5]	IRQCES[4]	IRQCES[3]	IRQCES[2]	IRQCES[1]	IRQCES[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-17. IUNIT IRQ Channel Enable Set Register (IRQn_IRQCES0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQCES	IRQ Channel Enable Set '0': No effect '1': Enables the corresponding IRQ channel This register always reads '0'. The bits in this register correspond to IRQ[31:0].

Note: IRQn_IRQCES0 provides channel enable set for IRQ source 0 to 31 (IRQ0 to IRQ31). As one register accommodates channel enable set for 32 sources, there exist 16 such registers (IRQn_IRQCES0 to IRQn_IRQCES15) to cover all IRQ sources (IRQ0 to IRQ511). IRQn_IRQCES1 provides channel enable set for IRQ source 32 to 63 (IRQ32 to IRQ63), IRQn_IRQCES2 provides channel enable set for IRQ source 64 to 96 (IRQ64 to IRQ95). The last register IRQn_IRQCES15 provides channel enable set for IRQ source 480 to 511 (IRQ480 to IRQ511).

17.2.17 IUNIT IRQ Channel Enable Clear Register (IRQn_IRQCEC0~15)

The software can use these registers to clear the individual IRQ channel enable in IRQn_IRQCE0~15 registers.

IUNIT IRQ Channel Enable Clear Register (IRQn_IRQCEC0)

Figure 17-18. IUNIT IRQ Channel Enable Clear Register (IRQn_IRQCEC0)

IRQn_IRQCEC0																															
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**
IRQCEC[31]	IRQCEC[30]	IRQCEC[29]	IRQCEC[28]	IRQCEC[27]	IRQCEC[26]	IRQCEC[25]	IRQCEC[24]	IRQCEC[23]	IRQCEC[22]	IRQCEC[21]	IRQCEC[20]	IRQCEC[19]	IRQCEC[18]	IRQCEC[17]	IRQCEC[16]	IRQCEC[15]	IRQCEC[14]	IRQCEC[13]	IRQCEC[12]	IRQCEC[11]	IRQCEC[10]	IRQCEC[9]	IRQCEC[8]	IRQCEC[7]	IRQCEC[6]	IRQCEC[5]	IRQCEC[4]	IRQCEC[3]	IRQCEC[2]	IRQCEC[1]	IRQCEC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-18. IUNIT IRQ Channel Enable Clear Register (IRQn_IRQCEC0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQCEC	IRQ Channel Enable Clear '0': No effect '1': Disables the corresponding IRQ channel This register always reads '0'. The bits in this register correspond to IRQ[31:0].

Note: IRQn_IRQCEC0 provides channel enable clear for IRQ source 0 to 31 (IRQ0 to IRQ31). As one register accommodates channel enable clear for 32 sources, there exist 16 such registers (IRQn_IRQCEC0 to IRQn_IRQCEC15) to cover all IRQ sources (IRQ0 to IRQ511). IRQn_IRQCEC1 provides channel enable clear for IRQ source 32 to 63 (IRQ32 to IRQ63), IRQn_IRQCEC2 provides channel enable clear for IRQ source 64 to 96 (IRQ64 to IRQ95). The last register IRQn_IRQCEC15 provides channel enable clear for IRQ source 480 to 511 (IRQ480 to IRQ511).

17.2.18 IUNIT IRQ Channel Enable Register (IRQn_IRQCE0~15)

The software can use these registers to enable/disable the corresponding IRQ channels. Software can also read the enable/disable status through these registers.

IUNIT IRQ Channel Enable Register (IRQn_IRQCE0)

Figure 17-19. IUNIT IRQ Channel Enable Register (IRQn_IRQCE0)

IRQn_IRQCE0																															
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**
IRQCE[31]	IRQCE[30]	IRQCE[29]	IRQCE[28]	IRQCE[27]	IRQCE[26]	IRQCE[25]	IRQCE[24]	IRQCE[23]	IRQCE[22]	IRQCE[21]	IRQCE[20]	IRQCE[19]	IRQCE[18]	IRQCE[17]	IRQCE[16]	IRQCE[15]	IRQCE[14]	IRQCE[13]	IRQCE[12]	IRQCE[11]	IRQCE[10]	IRQCE[9]	IRQCE[8]	IRQCE[7]	IRQCE[6]	IRQCE[5]	IRQCE[4]	IRQCE[3]	IRQCE[2]	IRQCE[1]	IRQCE[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-19. IUNIT IRQ Channel Enable Register (IRQn_IRQCE0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQCE	IRQ Channel Enable This register provides enabling/disabling of the IRQ channels. '0': Disables the corresponding IRQ channel '1': Enables the corresponding IRQ channel Reading provides enable/disable status of IRQ channels. The bits in this register correspond to IRQ[31:0].

Note: IRQn_IRQCE0 provides channel enable for IRQ source 0 to 31 (IRQ[0] to IRQ[31]). As one register accommodates channel enable for 32 sources, there exist 16 such registers (IRQn_IRQCE[0] to IRQn_IRQCE[15]) to cover all IRQ sources (IRQ[0] to IRQ5[11]). IRQn_IRQCE1 provides channel enable for IRQ source 32 to 63 (IRQ[32] to IRQ[63]), IRQn_IRQCE2 provides channel enable for IRQ source 64 to 96 (IRQ[64] to IRQ95). The last register IRQn_IRQCE15 provides channel enable for IRQ source 480 to 511 (IRQ[480] to IRQ[511]).

17.2.19 IUNIT NMI Hold Clear Register (IRQn_NMIHC)

The software can use this register to clear the hold bit corresponding to the current NMI ISR. The software writes NMI source number of the ISR.

IUNIT NMI Hold Clear Register (IRQn_NMIHC)

Figure 17-20. IUNIT NMI Hold Clear Register (IRQn_NMIHC)

IRQn_NMIHC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMIHCN[4]	NMIHCN[3]	NMIHCN[2]	NMIHCN[1]	NMIHCN[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-20. IUNIT NMI Hold Clear Register (IRQn_NMIHC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4:0]	NMIHCN	Hold Clear NMI Number IRQ source number corresponding to the NMI ISR. The read to this register field always returns '0'.

17.2.20 IUNIT NMI Hold Status Register (IRQn_NMIHS)

The software can use this register to read the hold status of individual NMI interrupts.

IUNIT NMI Hold Status Register (IRQn_NMIHS)

Figure 17-21. IUNIT NMI Hold Status Register (IRQn_NMIHS)

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

Table 17-21. IUNIT NMI Hold Status Register (IRQn_NMIHS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMIHS	NMI Hold Status '0': NMI[n] ISR is completed '1': NMI[n] is applied to the CPU

Note: IRQn_NMIHS provides hold status for NMI source 31 to 0 (NMI[31] to NMI[0]).

17.2.21 IUNIT IRQ Hold Clear Register (IRQn_IRQHC)

The software can use this register to clear the hold bit corresponding to the current IRQ ISR. The software writes IRQ source number of the ISR.

IUNIT IRQ Hold Clear Register (IRQn_IRQHC)

Figure 17-22. IUNIT IRQ Hold Clear Register (IRQn_IRQHC)

IRQn_IRQHC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IRQHCN[8]	IRQHCN[7]	IRQHCN[6]	IRQHCN[5]	IRQHCN[4]	IRQHCN[3]	IRQHCN[2]	IRQHCN[1]	IRQHCN[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-22. IUNIT IRQ Hold Clear Register (IRQn_IRQHC) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:9]	read0	-
[8:0]	IRQHCN	Hold Clear IRQ Number IRQ source number corresponding to the IRQ ISR. The read to this register field always returns '0'.

17.2.22 IUNIT IRQ Hold Status Register (IRQn_IRQHS0~15)

The software can use these registers to read the hold status of individual IRQ interrupt.

IUNIT IRQ Hold Status Register (IRQn_IRQHS0

Figure 17-23. IUNIT IRQ Hold Status Register (IRQn_IRQHS0)

IRQn_IRQHS0																															
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**
IRQHS[31]	IRQHS[30]	IRQHS[29]	IRQHS[28]	IRQHS[27]	IRQHS[26]	IRQHS[25]	IRQHS[24]	IRQHS[23]	IRQHS[22]	IRQHS[21]	IRQHS[20]	IRQHS[19]	IRQHS[18]	IRQHS[17]	IRQHS[16]	IRQHS[15]	IRQHS[14]	IRQHS[13]	IRQHS[12]	IRQHS[11]	IRQHS[10]	IRQHS[9]	IRQHS[8]	IRQHS[7]	IRQHS[6]	IRQHS[5]	IRQHS[4]	IRQHS[3]	IRQHS[2]	IRQHS[1]	IRQHS[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-23. IUNIT IRQ Hold Status Register (IRQn_IRQHS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQHS	IRQ Hold Status '0': IRQ[n] ISR is completed '1': IRQ[n] is applied to the CPU The bits in this register correspond to IRQ[31:0].

Note: IRQn_IRQHS0 provides hold status for IRQ source 0 to 31 (IRQ[0] to IRQ[31]). As one register accommodates hold status for 32 sources, there exist 16 such registers (IRQn_IRQHS0 to IRQn_IRQHS15) to cover all IRQ sources (IRQ[0] to IRQ[511]). IQRn_IRQHS1 provides hold status for IRQ source 32 to 63 (IRQ[32] to IRQ[63]), IRQn_IRQHS2 provides hold status for IRQ source 64 to 96 (IRQ[64] to IRQ[95]). The last register IRQn_IRQHS15 provides hold status for IRQ source 480 to 511 (IRQ[480] to IRQ[511]).

For IRQ source number details refer to device specific datasheet.

17.2.23 IUNIT IRQ Priority Level Mask Register (IRQn_IRQPLM)

The software can use this register to mask the IRQ sources based on their priority levels. All IRQ sources with greater or equal priority than the value programmed in this register will be masked. Default value of this register is 32, which mean all interrupts priority levels are active.

IUNIT IRQ Priority Level Mask Register (IRQn_IRQPLM)

Figure 17-24. IUNIT IRQ Priority Level Mask Register (IRQn_IRQPLM)

IRQn_IRQPLM																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IRQPLM[5]	IRQPLM[4]	IRQPLM[3]	IRQPLM[2]	IRQPLM[1]	IRQPLM[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 17-24. IUNIT IRQ Priority Level Mask Register (IRQn_IRQPLM) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:6]	read0	-
[5:0]	IRQPLM	IRQ Priority Level Mask These bits provide software priority level mask.

17.2.24 IUNIT Control and Status Register (IRQn_CSR)

The software can use this register to enable the IRQ processing block within the IUNIT, also this register provides IUNIT locked or unlocked status.

IUNIT Control and Status Register (IRQn_CSR)

Figure 17-25. IUNIT Control and Status Register (IRQn_CSR)

IRQn_CSR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	LST	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IRQEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-25. IUNIT Control and Status Register (IRQn_CSR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	LST	Interrupt Controller Lock Status Lock status for Interrupt Controller. '0': Interrupt Controller block is unlocked '1': Interrupt Controller block is still locked
[15:8]	read0	-
[7:1]	read0	-
[0]	IRQEN	Enable IRQ Block Interrupt Controller IRQ processing block enable. '0': IRQ processing block is disabled '1': IRQ processing block is enabled, **Note:** When disabled no new IRQs will be processed. Currently ongoing interrupt (if any) is still applied to the CPU.

17.2.25 IUNIT Nesting Level Status Register (IRQn_NESTL)

The software can use this register to read the nested interrupts status information.

IUNIT Nesting Level Status Register (IRQn_NESTL)

Figure 17-26. IUNIT Nesting Level Status Register (IRQn_NESTL)

IRQn_NESTL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IRQNL[4]	IRQNL[3]	IRQNL[2]	IRQNL[1]	IRQNL[0]	read0	read0	read0	NMINL[4]	NMINL[3]	NMINL[2]	NMINL[1]	NMINL[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R0	R0	R0	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-26. IUNIT Nesting Level Status Register (IRQn_NESTL) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:13]	read0	-
[12:8]	IRQNL	IRQ Nesting Level The number of IRQ ISR currently nested in the CPU.
[7:5]	read0	-
[4:0]	NMINL	NMI Nesting Level The number of NMI ISR currently nested in the CPU.

Note:

The NMINL and IRQNL contains the nesting levels of NMI and IRQ interrupt sources. The nesting level is determined by the number of ISR being nested inside the CPU. If the sum of NMI and IRQ ISR is greater than one, then it indicates there are nested interrupt.

If an IRQ interrupt is nested by an NMI interrupt then the IRQn_NESIL:NMINL is incremented, while the IRQn_NESIL:IRQNL holds the current count value.

The maximum value of IRQn_NESIL:IRQNL is 31, and maximum value of IRQn_NESIL:NMINL is 16.

17.2.26 IUNIT NMI Raw Status Register (IRQn_NMIRS)

The software can use this register to read the raw status of NMI sources.

IUNIT NMI Raw Status Register (IRQn_NMIRS)

Figure 17-27. IUNIT NMI Raw Status Register (IRQn_NMIRS)

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

Table 17-27. IUNIT NMI Raw Status Register (IRQn_NMIRS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMIRS	NMI Raw Status Each bit provides the raw status of corresponding NMI interrupt. '0': Interrupt is not set or cleared '1': Interrupt is set

Note: IRQn_NMIRS provides raw status of NMI source 31 to 0 (NMI[31] to NMI[0]).

17.2.27 IUNIT NMI Preprocessed Status Register (IRQn_NMIPS)

The software can use this register to read status of NMI sources before NMI processing stage.

IUNIT NMI Preprocessed Status Register (IRQn_NMIPS)

Figure 17-28. IUNIT NMI Preprocessed Status Register (IRQn_NMIPS)

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

Table 17-28. IUNIT NMI Preprocessed Status Register (IRQn_NMIPS) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NMIPS	NMI Preprocessed Status Each bit provides the status of corresponding NMI interrupts before NMI processing.

Note: IRQn_NMIPS provides status of NMI source 31 to 0 (NMI[31] to NMI[0]).

17.2.28 IUNIT IRQ Raw Status Register (IRQn_IRQRS0~15)

The software can use these registers to read the raw status of IRQ sources.

IUNIT IRQ Raw Status Register (IRQn_IRQRS0)

Figure 17-29. IUNIT IRQ Raw Status Register (IRQn_IRQRS0)

IRQn_IRQRS0																															
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**
IRQRS[31]	IRQRS[30]	IRQRS[29]	IRQRS[28]	IRQRS[27]	IRQRS[26]	IRQRS[25]	IRQRS[24]	IRQRS[23]	IRQRS[22]	IRQRS[21]	IRQRS[20]	IRQRS[19]	IRQRS[18]	IRQRS[17]	IRQRS[16]	IRQRS[15]	IRQRS[14]	IRQRS[13]	IRQRS[12]	IRQRS[11]	IRQRS[10]	IRQRS[9]	IRQRS[8]	IRQRS[7]	IRQRS[6]	IRQRS[5]	IRQRS[4]	IRQRS[3]	IRQRS[2]	IRQRS[1]	IRQRS[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-29. IUNIT IRQ Raw Status Register (IRQn_IRQRS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQRS	IRQ Raw Status Each bit provides the raw status of corresponding IRQ interrupt. '0': Interrupt is not set or cleared '1': Interrupt is set The bits in this register correspond to IRQ software interrupt IRQ[31:0].

Note: IRQn_IRQRS0 provides raw status of IRQ source 0 to 31 (IRQ[0] to IRQ[31]). As one register accommodates raw status of 32 sources, there exist 16 such registers (IRQn_IRQRS0 to IRQn_IRQRS15) to cover all IRQ sources (IRQ[0] to IRQ[511]). IRQn_IRQRS1 provides raw status of IRQ source 32 to 63 (IRQ[32] to IRQ[63]), IRQn_IRQRS2 provides raw status for IRQ source 64 to 96 (IRQ[64] to IRQ[95]). The last register IRQn_IRQRS15 provides raw status of IRQ source 480 to 511 (IRQ[480] to IRQ[511]).

17.2.29 IUNIT IRQ Preprocessed Status Register (IRQn_IRQPS0~15)

The software can use these registers to read status of IRQ sources before IRQ processing stage.

IUNIT IRQ Preprocessed Status Register (IRQn_IRQPS0)

Figure 17-30. IUNIT IRQ Preprocessed Status Register (IRQn_IRQPS0)

IRQn_IRQPS0																															
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**
IRQPS[31]	IRQPS[30]	IRQPS[29]	IRQPS[28]	IRQPS[27]	IRQPS[26]	IRQPS[25]	IRQPS[24]	IRQPS[23]	IRQPS[22]	IRQPS[21]	IRQPS[20]	IRQPS[19]	IRQPS[18]	IRQPS[17]	IRQPS[16]	IRQPS[15]	IRQPS[14]	IRQPS[13]	IRQPS[12]	IRQPS[11]	IRQPS[10]	IRQPS[9]	IRQPS[8]	IRQPS[7]	IRQPS[6]	IRQPS[5]	IRQPS[4]	IRQPS[3]	IRQPS[2]	IRQPS[1]	IRQPS[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-30. IUNIT IRQ Preprocessed Status Register (IRQn_IRQPS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IRQPS	IRQ Preprocessed Status Each bit provides the status of corresponding IRQ interrupt before IRQ processing. '0': Interrupt is not set or cleared '1': Interrupt is set The bits in this register correspond to IRQ interrupt IRQ[31:0].

Note: IRQn_IRQPS0 provides the status of IRQ sources 0 to 31 (IRQ0 to IRQ31). As one register accommodates the status of the 32 sources, 16 such registers (IRQn_IRQPS0 to IRQn_IRQPS15) exist to cover all IRQ sources (IRQ[0] to IRQ[511]). IRQn_IRQPS1 provides the status of IRQ sources 32 to 63 (IRQ[32] to IRQ[63]), IRQn_IRQPS2 provides status for IRQ sources 64 to 96 (IRQ[64] to IRQ[95]). The last register IRQn_IRQPS15 provides the status of IRQ sources 480 to 511 (IRQ[480] to IRQ[511]).

17.2.30 IUNIT Unlock Register (IRQn_UNLOCK)

The software can use this register to lock/unlock the IUNIT registers for write access.

IUNIT Unlock Register (IRQn_UNLOCK)

Figure 17-31. IUNIT Unlock Register (IRQn_UNLOCK)

IRQn_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-31. IUNIT Unlock Register (IRQn_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	IUNIT Unlock The IUNIT Unlock Register protects the IUNIT module from being modified accidently by software. The IUNIT registers cannot be written until this register has been written with a specific correct value. The correct value for unlocking can be written only in privilege mode. The read to this register always returns '0'. To lock the IUNIT again software must write another value specific to lock. Illegal access to IUNIT registers or writing value other than lock or unlock value to this register causes protection error. The list of registers in IUNIT which are not protected by lock or unlock are: * IRQn_NMIHC * IRQn_IRQHC * IRQn_IRQR0~IRQn_IRQR15 * IRQn_NMIR

17.2.31 IUNIT Module Identification Register (IRQn_MID)

This is a read-only register with a unique module identification number which identifies the version of the IUNIT module used in the MCU.

IUNIT Module Identification Register (IRQn_MID)

Figure 17-32. IUNIT Module Identification Register (IRQn_MID)

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

Table 17-32. IUNIT Module Identification Register (IRQn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	IUNIT Module ID The IUNIT module implemented in the device may vary from device to device. This register identifies the particular version of the hardware used in the device. This register helps in developing software to the hardware version implemented in the device. **Note:** For more details refer to the device specific datasheet for the version number details.

17.2.32 IUNIT ECC Error Interrupt Register (IRQn_EEI)

This register provides ECC error interrupts status and clear bits.

IUNIT ECC Error Interrupt Register (IRQn_EEI)

Figure 17-33. IUNIT ECC Error Interrupt Register (IRQn_EEI)

IRQn_EEI																													
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**
read0	read0	read0	read0	read0	read0	read0	EEIS	read0	read0	read0	read0	read0	read0	read0	EEIC	read0	read0	read0	read0	read0	read0	read0	EENS	read0	read0	read0	read0	read0	read0	read0	EENC
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-33. IUNIT ECC Error Interrupt Register (IRQn_EEI) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	EEIS	ECC Error IRQ Interrupt Status This register bit provides the status of ECC error IRQ interrupt. '0': Interrupt is not set or cleared '1': Interrupt is set This bit is set when single bit ECC error occurs on any read access to SRAM.
[23:17]	read0	-
[16]	EEIC	ECC Error IRQ Interrupt Clear This bit is used to clear the IRQ error interrupt. '0': No effect '1': Clears the IRQn_EEI:EEIS register bit Reading this bit always returns '0'.

Table 17-33. IUNIT ECC Error Interrupt Register (IRQn_EEI) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	EENS	ECC Error NMI Interrupt Status This register field provides the status of ECC error NMI interrupt. '0': Interrupt is not set or cleared '1': Interrupt is set This bit is set when double bits ECC error occurs on any read access to SRAM.
[7:1]	read0	-
[0]	EENC	ECC Error NMI Interrupt Clear This bit is used to clear the NMI error interrupt. '0': No effect '1': Clears the IRQn_EEI:EENS register bit Reading this bit always returns '0'.

Note: The IRQn_EEI:EENS should be cleared only after IRQ vector address re-initialization is done.

17.2.33 IUNIT ECC Address Number Register (IRQn_EAN)

The software can use this register to read the location of SRAM, where ECC error has occurred. This register is updated on read access to SRAM and single or double bit ECC error occurs.

IUNIT ECC Address Number Register (IRQn_EAN)

Figure 17-34. IUNIT ECC Address Number Register (IRQn_EAN)

IRQn_EAN																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	EAN[7]	EAN[6]	EAN[5]	EAN[4]	EAN[3]	EAN[2]	EAN[1]	EAN[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-34. IUNIT ECC Address Number Register (IRQn_EAN) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	EAN	ECC Address Number Address of SRAM location where ECC error has occured.

Note: Incase if double bit error occurs, and thereafter a single bit error occurs, then the IRQn_EAN register is not updated for single bit error until ECC NMI interrupt is cleared by writing '1' to IRQn_EEI:EENC register.

17.2.34 IUNIT ECC Test Register (IRQn_ET)

The software can use this register to enable ECC test functionality.

IUNIT ECC Test Register (IRQn_ET)

Figure 17-35. IUNIT ECC Test Register (IRQn_ET)

IRQn_ET																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ET
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-35. IUNIT ECC Test Register (IRQn_ET) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ET	ECC Test Enable '0': ECC test mode is disabled '1': ECC test mode is enabled

17.2.35 IUNIT ECC Error Bits Register (IRQn_EEB0 ~ 1)

The software can use this register for testing of ECC logic. Any bit in SRAM read data can be corrupted, by setting the corresponding bit of this register.

IUNIT ECC Error Bits Register (IRQn_EEB0)

Figure 17-36. IUNIT ECC Error Bits Register (IRQn_EEB0)

IRQn_EEB0																															
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**
EEB0[29]	EEB0[28]	EEB0[27]	EEB0[26]	EEB0[25]	EEB0[24]	EEB0[23]	EEB0[22]	EEB0[21]	EEB0[20]	EEB0[19]	EEB0[18]	EEB0[17]	EEB0[16]	EEB0[15]	EEB0[14]	EEB0[13]	EEB0[12]	EEB0[11]	EEB0[10]	EEB0[9]	EEB0[8]	EEB0[7]	EEB0[6]	EEB0[5]	EEB0[4]	EEB0[3]	EEB0[2]	EEB0[1]	EEB0[0]	read0	read0
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-36. IUNIT ECC Error Bits Register (IRQn_EEB0) bits

Bit Position	Bit Field Name	Bit Description
[31:2]	EEB0	ECC Error bits Bits in this register correspond to SRAM data [29:0].
[1:0]	read0	-

Note: The IRQn_EEB1 provides data corruption of SRAM data [66:37]

17.2.36 IUNIT ECC Error Bits Register (IRQn_EEB2)

The software can use this register to corrupt the computed 7 bits ECC check bits.

IUNIT ECC Error Bits Register (IRQn_EEB2)

Figure 17-37. IUNIT ECC Error Bits Register (IRQn_EEB2)

IRQn_EEB2																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	EEBO2[6]	EEBO2[5]	EEBO2[4]	EEBO2[3]	EEBO2[2]	EEBO2[1]	EEBO2[0]	read0	EEBE2[6]	EEBE2[5]	EEBE2[4]	EEBE2[3]	EEBE2[2]	EEBE2[1]	EEBE2[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-37. IUNIT ECC Error Bits Register (IRQn_EEB2) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15]	read0	-
[14:8]	EEBO2	ECC Error bits Bits in this register correspond to SRAM data [73:67].
[7]	read0	-
[6:0]	EEBE2	ECC Error bits Bits in this register correspond to SRAM data [36:30].

17.3 Interrupt Controller stages

The IUNIT provides a software interface to the interrupt system. The following sections explain the various stages within the IUNIT.

Interrupt raw status

The IUNIT provides raw status for IRQ and NMI interrupt sources. The raw status is the status of interrupts before set/reset stage.

Raw status for IRQ can be read through the IRQn_IRQRS0~IRQn_IRQRS015 registers.

Raw status for NMI can be read through the IRQn_NMIRS register.

Interrupt set/reset stage

The IUNIT supports software generated interrupts for both IRQ and NMI sources.

The software interrupt for IRQ can be set by writing '1' to the IRQn_IRQS0~IRQn_IRQS15 register bits

The software interrupt for IRQ can be cleared by writing '1' to the IRQn_IRQC0~IRQn_IRQC15 register bits.

The software interrupt status for IRQ can be read through the IRQn_IRQSS0~IRQn_IRQSS15 registers.

The software interrupt for NMI can be set by writing '1' to the IRQn_NMIS0 register bits

The software interrupt for NMI can be cleared by writing '1' to the IRQn_NMIC register bits.

The software interrupt status for NMI can be read through the IRQn_NMISIS register.

Interrupt enable/disable stage

The IUNIT supports enable/disable for IRQ sources. NMI sources cannot be disabled.

The IRQ source can be enabled by writing '1' to the IRQn_IRQCES0~IRQn_IRQCES15 register bits.

The IRQ source can be disabled by writing '1' to the IRQn_IRQCEC0~IRQn_IRQCEC15 register bits.

The IRQ source can also be enabled/disabled by writing directly to IRQn_IRQCE0~IRQn_IRQCE15 register bits.

The IRQ enable/disable status can be read through the IRQn_IRQCE0~IRQn_IRQCE15 registers.

Interrupt status

The IUNIT provides status for IRQ interrupts after enable/disable stage and status for NMI interrupts after set/reset stage.

Interrupt status for IRQ can be read through the IRQn_IRQST0~IRQn_IRQST015 registers.

Interrupt status for NMI can be read through the IRQn_NMIST register.

Hold clear stage

IUNIT keeps interrupt hold status for IRQ and NMI interrupts.

The hold bit for IRQ is set when the corresponding interrupt is applied to the CPU.

The software should clear the hold bit at the end of ISR. The hold bit for IRQ can be cleared by writing the IRQ source number to the IRQn_IRQHC register.

The hold status for IRQ interrupts can be read through the IRQn_IRQHS0~IRQn_IRQHS15 registers.

The hold bit for NMI is set when the CPU reads the IRQn_NMIVAS register, after nFIQ is applied to the CPU.

The software should clear the hold bit at the end of ISR. The hold bit for NMI can be cleared by writing the NMI source number to the IRQn_NMIHC register.

The hold status for NMI interrupts can be read through the IRQn_NMIHS register.

The hold status provides information of currently active interrupts.

IRQ processing

The IRQ sources are processed based on their software as well as hardware priorities and then highest priority interrupt source is applied to the CPU.

The software priorities for IRQ sources can be set through IRQn_IRQPL0~IRQn_IRQPL127 registers. The IUNIT provides 32 levels of software priorities.

The hardware priorities are fixed priority. The lower the interrupt source number, the higher the hardware priority.

The IUNIT also provides masking of software priorities through software level mask register IRQn_IRQPLM.

NMI processing

The NMI sources are processed based on their software as well as hardware priorities and then highest priority interrupt source is applied to the CPU.

The software priorities for NMI sources can be set through IRQn_NMIPL0~IRQn_NMIPL16 registers. The IUNIT provides 16 levels of software priorities.

The NMI source 0 is the highest priority interrupt. Hence software priority for NMI source 0 is fixed to 0 and it cannot be changed.

The hardware priority is fixed priority. The lower the interrupt source number, the higher the hardware priority.

17.4 Notes for software

The following section describes the notes for software.

- The software interrupt should be cleared by its own ISR
- If two interrupts with the same programmed priority level are asserted simultaneously, the interrupt connected to the lowest IRQ source is serviced first. If an interrupt occurs, while another interrupt of the same priority level is being serviced, it is masked until the first interrupt service routine has completed
- All ISR vector addresses must be programmed via software during initialization
- For proper operation of IUNIT, the interrupt flag at peripheral must be cleared first before clearing the corresponding hold bit in IUNIT
- For software interrupts, the software interrupt set bit in IUNIT must be cleared first before clearing the corresponding hold bit in IUNIT
- In case of double bit ECC errors in SRAM, the corresponding NMI handler must appropriately re-initialize the SRAM locations
- Software while in main program, sees any of the hold bit set must take appropriate action

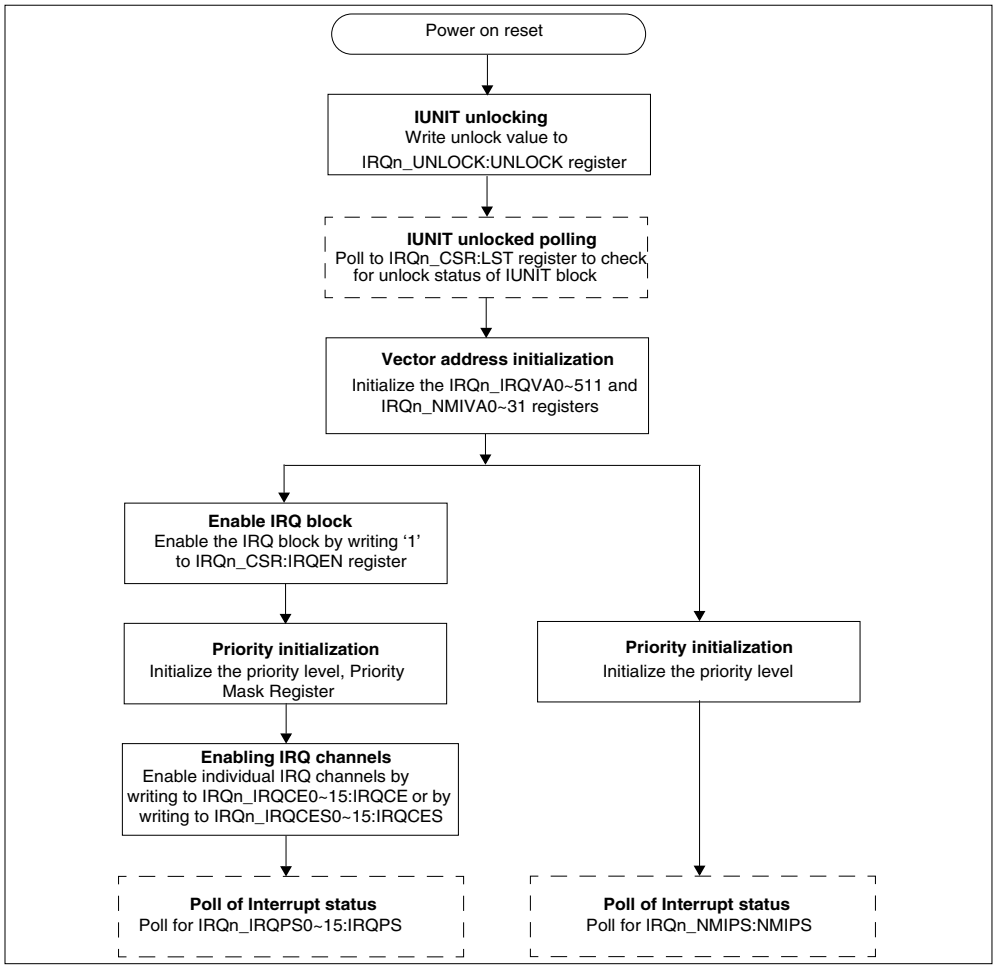
17.5 Operations flowchart

IUNIT initialization and nesting operation is explained in this section.

Initialization flowchart

The IUNIT initialization procedure is shown in Figure 17-38.

Figure 17-38. Interrupt process flowchart



After power on reset sequence, the IUNIT block needs to be unlocked for performing a write access. This is done by writing an 'unlock' value in the IRQn_UNLOCK:UNLOCK register. This unlocks the IUNIT block for writing. The software can perform a read of IRQn_CSR:LST bit poll for the IUNIT block unlocked status. This bit will be made 0 once the IUNIT is unlocked.

The vector addresses for all interrupt sources (NMI and IRQ) needs to be initialized. Also, the priority level for all interrupt sources needs to be programmed respectively. The IRQ processing block will be enabled on writing '1' to the IRQn_CSR:IRQEN bit. Also, each of the IRQ channels needs to be enabled by writing '1' to the IRQn_IRQCE0~15 register bits. By default all IRQ channels are disabled.

Based on the priority computation the highest priority interrupt will be applied to the CPU on the nFIQ (NMI), nIRQ (IRQ) lines.

The currently active interrupt status can be read by software through the following registers IRQn_IRQVA, IRQn_IRQST for nIRQ interrupt, and IRQn_NMIVAS, IRQn_NMIST for nFIQ interrupt. The nested interrupt status can be read on IRQn_NESTL register.

To disable the IRQ processing block at any time the IRQn_CSR:IRQEN bit can be written '0', on this any ongoing interrupt will be allowed to complete and there after the IRQ processing block actually gets disabled.

If software needs to set any of the interrupt source independently, it can write '1' to the IRQn_IRQS0~15:IRQS register bit for nIRQ and IRQn_NMIS:NMIS register bit for nFIQ. Each of the ISR needs to clear the software interrupt by writing '1' to IRQn_IRQR0~15:IRQR register bit for nIRQ and IRQn_NMIR:NMIR register bit for nFIQ.

Note: The 'unlock and lock' values are provided in the device specific datasheet.

Interrupt/NMI nesting

The Cortex R4F register set does not handle nested interrupts automatically. This needs to be done in an application within the Interrupt Service Routine (ISR).

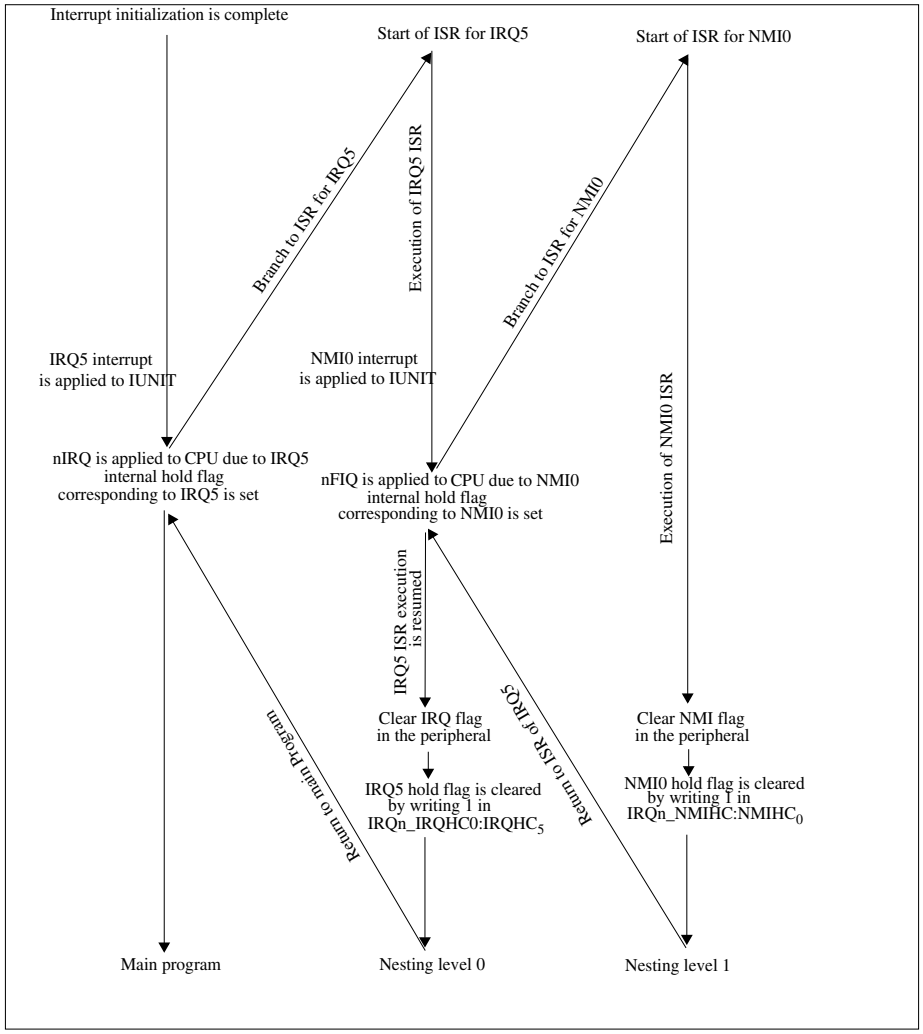
See also Arm Cortex-4F Technical Reference Manual for interrupt save and restore behavior.

Interrupt and NMI behavior without using special software nesting sequence

Without adding these software handling the first active interrupt request will be executed and next incoming interrupt requests need to wait until active ISR is finished. The same applies to NMI handling. This Interrupt Controller is using nIRQ line for 'normal' interrupts and nFIQ for NMI. So NMI request having higher hardware priority than interrupts and can interrupt them.

Nesting operation of NMI and IRQ

Figure 17-39. Interrupt nesting process flow



When the interrupt source (IRQ5) is applied to the IUNIT, the IUNIT will process the interrupt to generate nIRQ to the CPU. The IUNIT will also set the internal hold flag corresponding to IRQ5. The CPU stops current state, stacks the current workspace, reads the vector address of IRQ5, branches to the ISR address, and starts executing the ISR. During this time the NMI interrupt source (NMI0) is applied to the IUNIT and the IUNIT will process the interrupt to generate nFIQ to the CPU. The IUNIT will also set the internal hold flag corresponding to NMI0. As the NMI interrupts have higher priority inside CPU, the CPU stops the current IRQ5 ISR and stores the workspace, reads the vector address of NMI0, branches to the ISR address, and starts executing the ISR. Here nesting of interrupts is observed. Before completion of the ISR for NMI0, the interrupt is cleared in the peripheral by the ISR, also the hold bit which is set earlier is cleared by the ISR. Once the ISR for NMI0 is finished the CPU resumes the ISR for IRQ5 from the state it had left earlier in time. Before completion of the ISR for IRQ5, the interrupt is cleared in the peripheral by the ISR, also the hold bit which is set earlier is cleared by the ISR. On completion of the ISR for IRQ5, the CPU returns back to the main program.

If the interrupt source is generated via software by writing '1' to the particular register bit of IRQ Set Register (IRQn_IRQS0~15)/NMI Set Register (IRQn_NMIS). Then the ISR should clear the software generated interrupt by writing '1' to the corresponding register bit of IRQ Reset Register (IRQn_IRQR0~15)/NMI Reset Register (IRQn_NMIR). This should be done before clearing the hold bit of the corresponding interrupt.

Note: VIC needs to be configured prior to usage. Refer to Arm Cortex-R4 TRM DDI0363C.

Interrupt/NMI priorisation methods

The Interrupt Controller support level priorisation, 32 levels can be assigned to the interrupts. In addition there is a hardware priorisation, the lowest interrupt number has highest priority. In case two or more interrupts of same level occurs at same time the lowest hardware number will execute first. In case an interrupt is already executed, an interrupt using same level but higher hardware priority will not interrupt current interrupt execution.

The same applies to NMI. NMI0 has fixed priority 0 (highest), for all other available NMI sources priority can be set according to application needs.

The actual IRQ priority level mask is shown in IRQn_IRQST:IRQPS[4:0] register bits and the actual NMI priority level mask is shown in IRQn_NMIST:NMIPS[3:0] register bits.

Figure 17-40. ISR execution flow for no nested interrupts

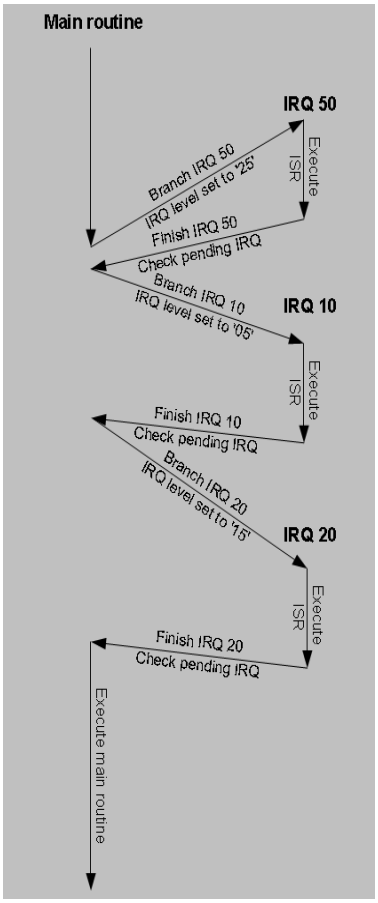


Figure 17-40 shows a scenario where the following three interrupts are initialized.

Interrupt	Priority level
IRQ10	5
IRQ20	15
IRQ50	25

Interrupts are initialized and enabled, VIC port is enabled and interrupt level is default (32). IRQ50 interrupt request occurs, as no other interrupt is pending IRQ50 is executed, and level is set to 25 (IRQn_IRQST:IRQPS[4:0]). During the execution of IRQ50 interrupt service routine interrupt request for IRQ20 level is set to 15 (IRQn_IRQST:IRQPS[4:0]) and then IRQ10 occurs and interrupt level is set to 5 (IRQn_IRQST: IRQPS[4:0]). As IRQ50 is not finished, these request need to wait until interrupt IRQ50 processing is finished. After finalization of IRQ50 there are the above mentioned pending interrupts. As IRQ10 is having higher priority (level 5) than IRQ20 it is executed next. After IRQ 10 is finished IRQ 20 is executed and IRQ priority level mask is set to 15 (IRQn_IRQST: IRQPS[4:0]). After end of IRQ20 return to application main routine execution as no other IRQ are pending and IRQ priority level mask is set to 32 (IRQn_IRQST:IRQPS[4:0]).

Interrupt behavior using special software nesting sequence

If nesting of interrupts shall be used some software needs to be added into the Interrupt Service Routine of each interrupt.

Figure 17-41. ISR execution flow for nested interrupts

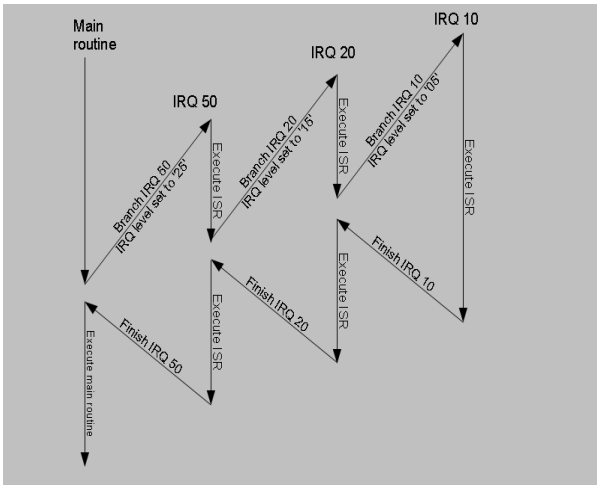


Figure 17-41 shows a scenario using nested interrupt functionality. Three interrupts are initialized:

Interrupt	Priority level
IRQ10	5
IRQ20	15
IRQ50	25

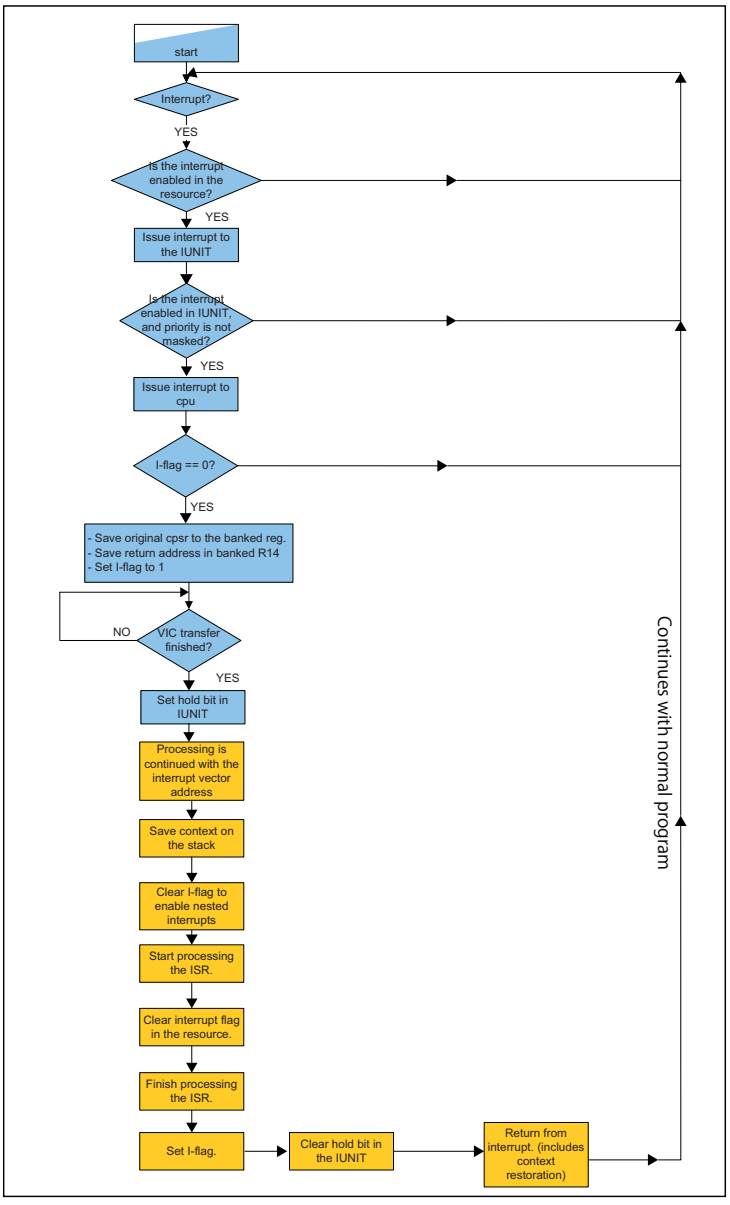
Interrupts are initialized and enabled, VIC port is enabled, and interrupt level is default (32). IRQ50 interrupt request occurs, as no other interrupt is pending IRQ50 is executed and IRQ priority level mask is set to 25 (IRQn_IRQST:IRQPS[4:0]). During the execution of IRQ50 interrupt service routine interrupt a request for IRQ20 occurs. The IRQ priority level mask is set to 15 (IRQn_IRQST:IRQPS[4:0]), IRQ50 execution is stopped and IRQ20 ISR is executed. The value of IRQn_NESTL:IRQNL[4:0] bits in IUNIT Nesting Level Status Register (IRQn_NESTL) is set from value 0 to 1 (one nested IRQ). During this execution an IRQ10 interrupt request occurs. The IRQ priority level mask is set to 5 (IRQn_IRQST:IRQPS[4:0]), IRQ20 processing is stopped, and IRQ10 ISR is executed.

The value of IRQn_NESTL:IRQNL[4:0] is set to value 2. After IRQ10 is finished, IRQ20 will be continued and IRQ priority level mask is set to 15 and IRQn_NESTL:IRQNL[4:0] is set to 1. After end of IRQ20, IRQ50 will be continued and IRQ priority level mask set to 25 and after end of IRQ50 IRQ priority level mask is set to 32, IRQn_NESTL:IRQNL[4:0] is set to 0 and jump back to application main routine.

Nested interrupt software example

As the Arm architecture does not support nested interrupts for nIRQ, the application needs to take care of it. Within the ISR the LR_irq and SPSR_irq register values needs to be stored at system stack and interrupts enabled again. Now any new IRQ request (if priority is higher than actual ISR) is able to interrupt current ISR execution. At end of ISR the saved LR_irq and SPSR_irq register values needs to be restored from system stack.

Figure 17-42. IRQ Flow Diagram



Software example

IRQxx_Handler

```
SUB     lr, lr, #4          ; Save LR_irq and SPS_irq on system stack
SRSDB   #0x1f!
CPS     #0x1f               ; switch to system stack
PUSH    {r0-r3, r12}        : store remaining AAPCS register on system stack
AND     r1, sp, #4          ; ensure stack is 8-byte aligned
```

```
SUB         sp, sp, r1
PUSH        {r1, lr}
CPSIE       i                       ; enable interrupts
...         ; ISR code which can be interrupt be other high prior interrupts
; end of ISR:
CPSID       I                       ; disable IRQ
POP         {r1, lr}                ; restore LR_sys and unadjust stack
ADD         sp, sp, r1
POP         {r0-r3, r12}            ; restore AAPCS register
RFEIA       SP!                     ; return from system stack mode
```

Note: When using nesting of interrupts and/or NMI the additional required system stack size have to be calculated according to used level depth.

Nested NMI handling

NMI processing

In FCR4 Cluster series devices the fast interrupt of the core is configured as NMI. Also for NMIs the flag is located in the source module. The difference to the IRQs is, that there does not exist an enable bit, neither in the resource nor in the IUNIT.

The NMI entry is as follows:

1. The interrupt flag is set by HW and it is issued to the Interrupt Controller (IUNIT).
2. In the IUNIT it is done a priority evaluation. If the priority of this NMI is the highest, the NMI is issued to the CPU.
3. The CPSR is copied to the SPSR_fiq and the r14_fiq gets the return address. Then in the CPSR the mode is set to fiq and the I-flag and the F-flag is set (this disables the NMIs). All this is done in HW.
4. When the processor takes this interrupt, there should be read out the register in the IUNIT (mirror near the BootRom) which holds the resulting vector address of this NMI. With this read access the hold bit of the regarding NMI is set. This read is done by the BootRom code (instruction at FIQ entry).
5. The processing continues now with the address which was read out of the IUNIT register.
6. Because the system should support nested NMIs the F-flag should be cleared afterwards (SW).
7. Before clearing the F-flag, the context must be saved to the stack (SW).
8. After clearing the F-flag an NMI with higher priority can interrupt the current NMI again.

The SW part which must be done (with a disabled F-flag) is the following. This is just an example, maybe other instructions can be used:

- Reading the resulting vector address from the IUNIT (bus access)
- SRS (Store Return State), this instruction stores the r14_fiq and the SPSR_fiq on the stack. (address is specified through the r13_fiq). The r13_fiq is also updated accordingly
- STM (Store Multiple), this instruction stores a defined number of registers (r0-r14) to the stack defined by r13_fiq
- Update r13_fiq accordingly (not available in STM)
- Clear F-flag (enable nested NMIs)
- Process ISR

During all the initial instructions for context saving the F-flag is set, so these procedure can not become interrupted by another NMI.

1. The interrupt exit is as follows: During the ISR the NMI flag in the source must be cleared by SW. This should be done at a possibly early time in the ISR (clock synchronization).
2. At the end of the ISR the hold bit in the Interrupt Controller should be cleared, by writing the NMI number to a special register in the Interrupt Controller (refer to IUNIT spec). When this write operation is done, the active status is removed from this NMI in the IUNIT. This is also done by SW.

3. When this is done, the return from interrupt and the restoration of the stacked context including the PC, CPSR must be done (SW). The restoration of the CPSR automatically enables also the normal interrupts again, because it restores the status of the register directly before the NMI occurred.

Here the SW part is the following:

- Clear hold bit in the Interrupt Controller by writing to the NMI number to a certain register
- LDM (exception return), this instruction restores the saved user registers and the CPSR and the PC from the stack. The r13_fiq is updated accordingly

Through restoration of CPSR the mode changes back to user mode and the I-flag is cleared implicitly.

In this sequence a problem can occur, in case the hold bit is cleared, and another NMI with lower or equal priority is still pending in the IUNIT. This pending NMI is immediately after clearing the hold bit of the previous one issued to the CPU.

If the LDM instruction of the first NMI is delayed (e.g. because of an ECC error while instruction fetch) it can happen, that the second NMI is issued to the CPU before executing the LDM instruction of the first one.

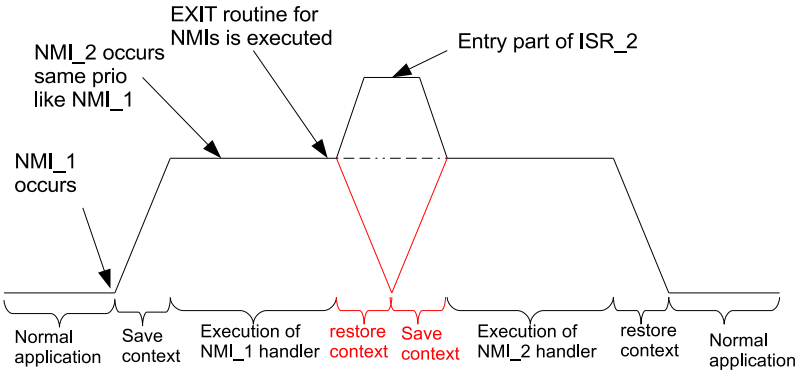
The effect is, that the stacked context of the previous NMI are still saved on the stack. Now the ISR of the second NMI saves again the whole context to the stack. Effectively there are now the same context saved two times on the stack.

To solve this problem there should be a common exit routine for NMIs (EXIT_NMI). In this exit routine two things should be done:

- Clear hold bit in Interrupt Controller (one instruction)
- Return from interrupt and restore context (one instruction)

This is shown in the following figure.

Figure 17-43. Behaviour in case of enabled nested NMIs

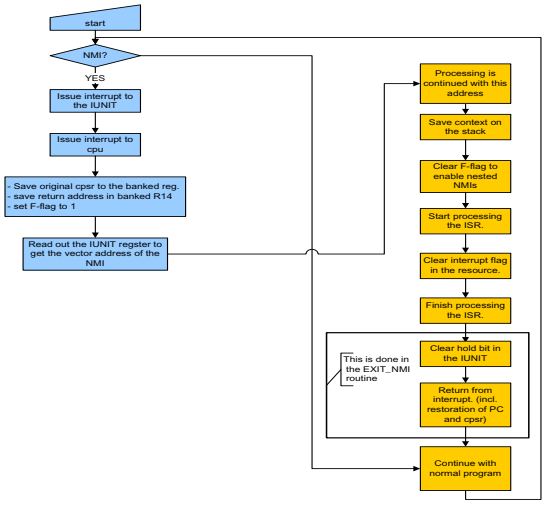


At the entry of each NMI routine there are some registers saved on the stack. After that it is compared, whether the PC before the NMI was equal to the 'return from NMI' instruction in the EXIT_NMI routine.

If this is the case, this is the information, that an NMI was interrupted between the two instructions in the EXIT_NMI routine. Then no stacking of the context is required. If not, the full context must be saved.

The following figure shows the flow of an NMI.

Figure 17-44. NMI flow



Hardware generated software interrupts

This Interrupt Controller supports 512 IRQ register. Depending on series not all of this registers are connected to a hardware interrupt source. These registers can be used for additional software interrupts or test of hardware interrupt ISR execution. Through IRQn_IRQS0~15 register it is possible to issue an interrupt request by writing '1' at the corresponding bit in the register. Within the ISR this interrupt request needs to be reset by writing a '1' to the corresponding bit position in the IRQn_IRQR0~15 register.

Note: Refer to the series datasheet for IRQ assignments.

ECC support and test

This Interrupt Controller also uses SRAM, which can be used with ECC protection. By default ECC protection is activated for this Interrupt Controller SRAM.

It is possible to generate ECC failure in SRAM area via the registers IRQn_EEB0, IRQn_EEB1, and IRQn_EEB2. These register can be used to check ECC error handler e.g. during development of application. When writing a '1' to a dedicated bit(s) in one or more of IRQn_EEB0, IRQn_EEB1, and IRQn_EEB2 register, this bit position will be mask (stipulate error). A read access to Interrupt Controller RAM will generate an interrupt in case of 1-bit (correctable) error of ECC NMI request in case of >1-bit (uncorrectable) error.

18. Port Pin Configuration

This chapter explains the functions and operations of the Port Pin Configuration module.

18.1 Outline of Port Pin Configuration

This section describes the features and the block diagram of the Port Pin Configuration.

Features of Port Pin Configuration

The Port Pin Configuration module controls input and output data transfer through the port pins. It implements multiplexing functionality to route the selected resource output to its associated port pin. It also provides the controls needed for the operation of the IO cell.

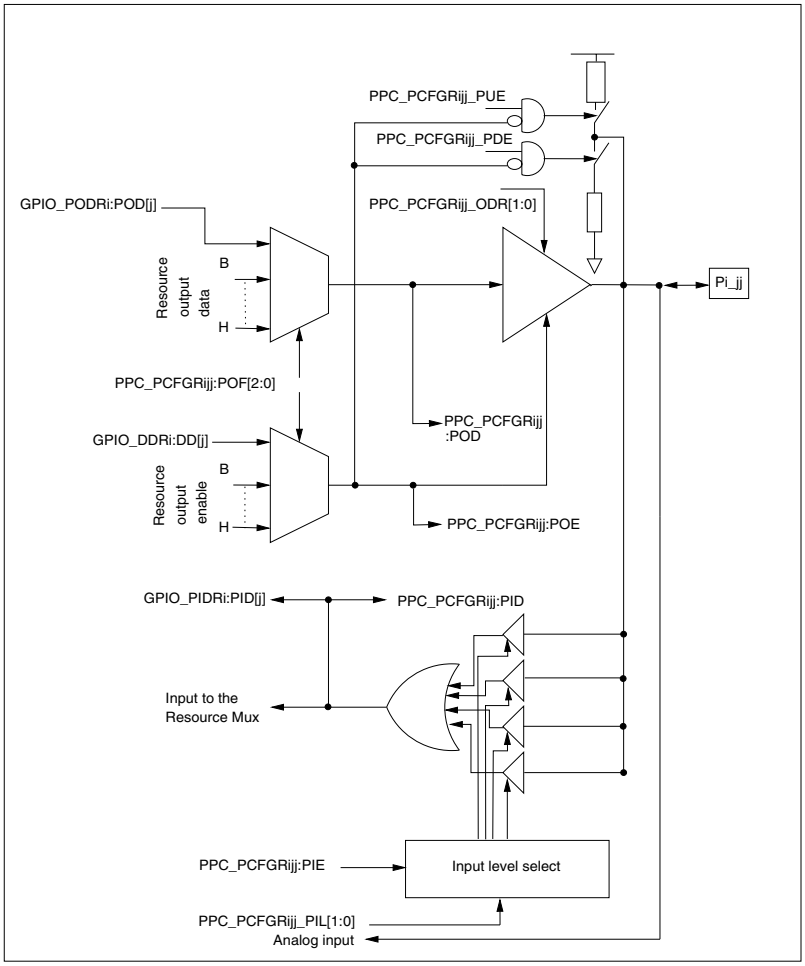
The features of the Port Pin Configuration module are as follows:

- A 16-bit Pin Configuration Register (PPC_PCFGRijj) is associated with each programmable port pin Pi_jj, where i=0..7, jj=00..63 (two digits) and j = 0..63 (no leading zero)
- PPC_PCFGRijj also comprises status bits for output data, output enable, and pin input data of the IO cell
- Port pin functionality is selected by programming the PPC_PCFGRijj:POF[2:0] register bits

Block diagram of Port Pin Configuration

Figure 18-1 shows the Port Pin Configuration block diagram. Each output port pin can be driven either by the GPIO_PODRi:POD[j] bit in the GPIO modul or by one of the resources sharing the port pin. Up to three port pin output function select bits (PPC_PCFGRijj:POF[2:0]) associated with each port pin select the output function through the port pin. Output enable for the pin is controlled by the respective resource/GPIO. When the port pin is configured as input, the PPC_PCFGRijj:PID bit indicates the input data.

Figure 18-1. Block diagram of Port Pin Configuration



18.2 Port Pin Configuration registers

This section describes the registers of the Port Pin Configuration module in detail.

Registers of Port Pin Configuration

The following registers are available for Port Pin Configuration:

Pin Configuration Register (PPC_PCFGRijj)

The suffix 'ijj' in the register name indicates that the register corresponds to the port pin Pi_jj of the device.

Memory layout of the Port Pin Configuration registers

For each configurable port pin, there is a corresponding 16-bit PPC_PCFGRijj. Dedicated port pins, such as RSTX and MODE do not have an associated PPC_PCFGRijj. The PPC_PCFGRijj memory map for 512 port pins is shown in Table 18-1. The registers can be accessed 8-bit or 16-bit wide.

Table 18-1. Memory layout of Port Pin Configuration registers

Offset	+1	+0
0x00000000 to 0x0000007F	PPC_PCFGR000~063 0XX0000000000000	
0x00000080 to 0x000000FF	PPC_PCFGR100~163 0XX0000000000000	
0x00000100 to 0x0000017F	PPC_PCFGR200~263 0XX0000000000000	
0x00000180 to 0x000001FF	PPC_PCFGR300~363 0XX0000000000000	
0x00000200 to 0x0000027F	PPC_PCFGR400~463 0XX0000000000000	
0x00000280 to 0x000002FF	PPC_PCFGR500~563 0XX0000000000000	
0x00000300 to 0x0000037F	PPC_PCFGR600~663 0XX0000000000000	
0x00000380 to 0x000003FF	PPC_PCFGR700~763 0XX0000000000000	

18.2.1 Pin Configuration Register (PPC_PCFGRijj)

Each port pin has a 16-bit Pin Configuration Register. Bits[15:14] are associated with output status, bit 13 with input status, bits [12:8] with input configuration, and the lower 8 bits with output configuration.

Pin Configuration Register (PPC_PCFGRijj)

Figure 18-2. Pin Configuration Register (PPC_PCFGRijj)

PPC_PCFGRijj															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
POE	POD	PID	PIE	PIL[1]	PIL[0]	PUE	PDE	ODR[1]	ODR[0]	read0	read0	read0	POF[2]	POF[1]	POF[0]
Rp	Rp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	**X**	**X**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Note: PPC_PCFGRijj:PIL[0] = '0' for 3 V IO cell and '1' for 5 V IO cell.

Table 18-2. Pin Configuration Register (PPC_PCFGRijj) bits

Bit position	Bit name	Description
15	POE	Pin Output Enable This read-only bit indicates the status of the pin output enable. '0': Output is Hi-Z '1': Output is enabled
14	POD	Pin Output Data This read-only bit indicates the output data driven to the pin. POD is meaningful only if PPC_PCFGRijj:POE = '1': PPC_PCFGRijj:POE = '0', POD = Don't care, Output is Hi-Z PPC_PCFGRijj:POE = '1', POD = '0', Output is driven to '0' PPC_PCFGRijj:POE = '1', POD = '1', Output is driven to '1' Note: The POD bit has no meaning when a pin of IO circuit type RSDS is put in RSDS mode (this is done by selecting the resource that is using the RSDS mode by the PPC_PCF-GRijj:POF[2:0] bits).
13	PID	Pin Input Data This read-only bit indicates the input data detected at the input buffer selected by PPC_PCFGRijj:PIL[1:0]. If PPC_PCFGRijj:PIE = '0', this bit returns X.

Table 18-2. Pin Configuration Register (PPC_PCFGRijj) bits

Bit position	Bit name	Description
12	PIE	Pin Input Enable Set this bit to '1' to enable digital input buffers. Ensure to set this bit to '0', when the pin is left open, or when analog voltage is applied to the pin. Note: For pins with an ADC input, the input buffer is disabled irrespective of the PIE value if the corresponding ADC channel is enabled, i.e. if the corresponding bit of the ADCn_ER32/ ADCn_ER10 register is set to ’1’.
[11:10]	PIL	Pin Input Level These bits select the programmable input level. Refer to the IO Circuit Types section in the device specific datasheet for the information how to configure these bits for the different IO circuit types.
9	PUE	Pull-Up Enable Set this bit to '1' to enable pull-up resistor in input mode.
8	PDE	Pull-Down Enable Set this bit to '1' to enable pull-down resistor in input mode.
[7:6]	ODR	Pin Output Drive These bits select the programmable output drive. Refer to the IO Circuit Types section in the device specific datasheet for the information how to configure these bits for the different IO circuit types.
[5:3]	read0	
[2:0]	POF	Port Pin Output Function Select These bits select the output function of the port pin. '000': General purpose port output (GPIO_PODRi:POD[j]) '001': Resource function B output '010': Resource function C output '011': Resource function D output '100': Resource function E output '101': Resource function F output '110': Resource function G output '111': Resource function H output Refer to the Port Pin Multiplexing table in the device specific datasheet for the information which resources are available at which pins.

18.3 Notes on using Port Pin Configuration

This section describes the register protection feature of Port Pin Configuration.

Register protection feature

All Pin Configuration registers PPC_PCFGRijj are protected by the PPU channels PPCUSER (in user mode) and PPCPRIV (in privileged mode). In addition, each Pin Configuration Register is individually protected by its corresponding GPIO_PPERi:PPE[j] register bit.

Writing to the PPC_PCFGRijj register either in user mode or privileged mode is permitted only if the access permission bit of the corresponding PPU channel (PPCUSER for user mode, PPCPRIV for privileged mode) is set to '1' and the GPIO_PPERi:PPE[j] bit is set to '1'.

Similarly, reading the PPC_PCFGRijj register either in user mode or privileged mode is permitted only if the read permission bit or the access permission bit of the corresponding PPU channel (PPCUSER for user mode, PPCPRIV for privileged mode) is set to '1' and the GPIO_PPERi:PPE[j] bit is set to '1'.

Refer to the PPU section of the device specific datasheet for the location of the PPCUSER and PPCPRIV PPU channels.

It can happen that in cases of read and write permission violations, the Bus Error Collection Unit is not signalled, meaning no error response will be generated.

Table 18-3. Required permission for access to PPC_PCFGRijj

Access type	PPU read permission in user mode	PPU access permission in user mode	PPU read permission in privileged mode	PPU access permission in privileged mode	GPIO_PPERi: PPE[j]
No access	X	X	X	X	'0'
No access in user mode	'0'	'0'	X	X	'1'
In user mode, only read access is permitted	'1'	'0'	X	X	'1'
In user mode both read and write access is permitted	X	'1'	X	X	'1'
No access in privileged mode	X	X	'0'	'0'	'1'
In privileged mode, only read access is permitted	X	X	'1'	'0'	'1'
In privileged mode, read and write access is permitted.	X	X	X	'1'	'1'

Note: It is not recommended to enable both PPC_PCFGRijj:PDE and PPC_PCFGRijj:PUE at the same time as this results in increased current consumption.

Caution.

- If a pin is left open or an analog voltage applied to it, then the pin's digital input buffer should be disabled by setting the corresponding PPC_PCFGRijj:PIE bit to '0' to avoid excess current consumption
- If a pair of IOs is used for graphics display, then both pins must be selected for graphics output
- If an NMI port is enabled in the Resource Mux Configuration Register, an output driver is disabled (PPC_PCFGRijj:POE = '0') and an input enabled (PPC_PCFGRijj:PIE = '1') for that port for safety. Other port settings (e.g. pull up/down, input level) are not affected by the NMI enable bit. The user must configure the port (i.e. input levels) before and lock it by writing the GPIO_PPERi:PPE[j] bit to '0' before enabling the NMI port. For more details refer to 35. GPIO Module

19. External Interrupt Capture Unit

This chapter explains the function and operation of the External Interrupt Capture Unit (EICU).

19.1 Outline of the External Interrupt Capture Unit (EICU)

This section describes the features and the block diagram of EICU module.

Features of External Interrupt Capture Unit (EICU)

The EICU module samples the interrupt pin level in isochronous steps after being triggered by an External Interrupt event on that pin. It can be used to record the bit stream that causes the External Interrupt event like CAN wakeup. Features of the EICU are:

- It supports the capture of events among 32 External Interrupt Pins (Pin INTk)
- The capture on any interrupt pin is maskable
- It has a 6-bit linearly programmable prescaler to provide a sample frequency range of 500 Hz to 16 MHz
- Software can read the sample registers and enable for the next observation
- Software can also read the channel number being observed
- It supports the Peripheral Protection Unit (PPU)

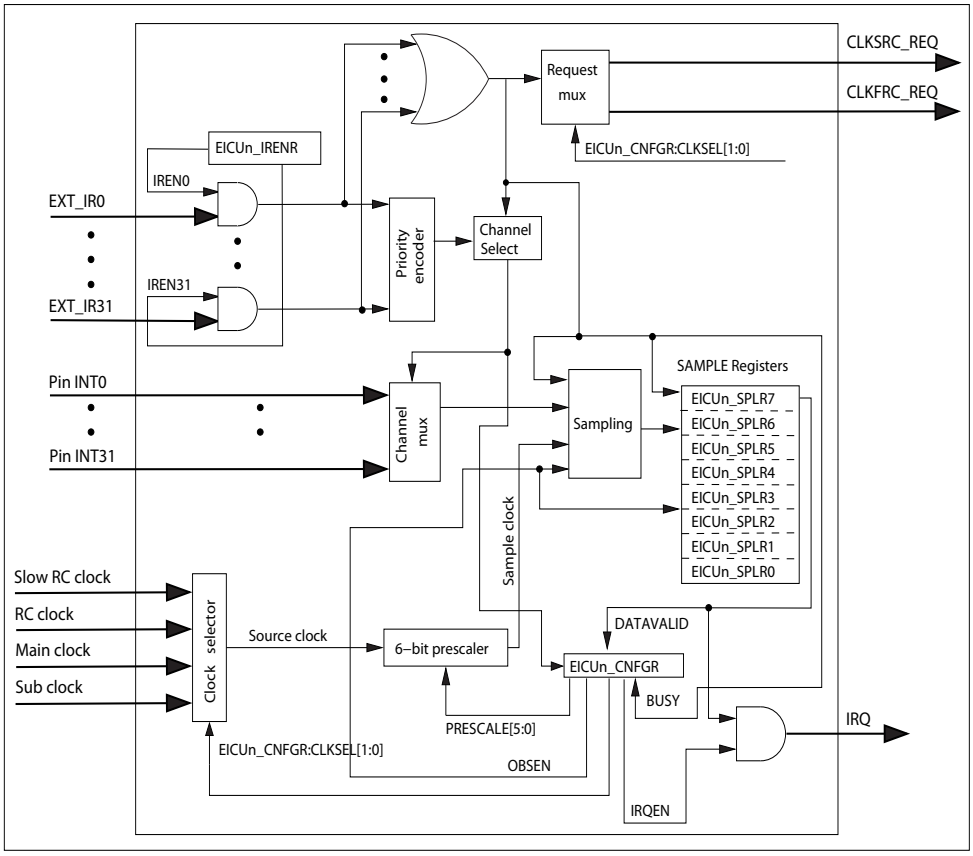
DMA and interrupts

This module does not generate any DMA requests.

An interrupt is generated when valid data is available in the sample registers.

Block diagram of EICU

Figure 19-1. Block diagram of EICU



19.2 EICU registers

The EICU module has various registers that enable the channels for observation, configure the prescaler and source clock, and store the sample data. It also has a status register to update the data validity status and the channel number under observation. All registers in the EICU module are explained in this section.

The suffix 'n' in the register name indicates that the register is in instance 'n' of the module.

Registers of EICU

The following registers are available for each instance of EICU:

- Configuration Register (EICUn_CNFGR)
- External Interrupt Pin Enable Register (EICUn_IRENR)
- Sample Registers 0~7 (EICUn_SPLR0~7)

Memory layout of EICU registers

Table 19-1. Memory layout of the EICU registers

Offset	+3	+2	+1	+0
0x00000000	EICUn_CNFGR 00000000 00000000 00000000 00000000			
0x00000004	EICUn_IRENR 00000000 00000000 00000000 00000000			
0x00000008	EICUn_SPLR0 00000000 00000000 00000000 00000000			
0x0000000C	EICUn_SPLR1 00000000 00000000 00000000 00000000			
0x00000010	EICUn_SPLR2 00000000 00000000 00000000 00000000			
0x00000014	EICUn_SPLR3 00000000 00000000 00000000 00000000			
0x00000018	EICUn_SPLR4 00000000 00000000 00000000 00000000			
0x0000001C	EICUn_SPLR5 00000000 00000000 00000000 00000000			
0x00000020	EICUn_SPLR6 00000000 00000000 00000000 00000000			
0x00000024	EICUn_SPLR7 00000000 00000000 00000000 00000000			

19.2.1 Configuration Register (EICUn_CNFGR)

This register is used to configure the module for its prescaler value and source clock. It also gives informtion concerning the sampling status, data validity in the Sampling Registers (SPLR) and the current interrupt pin being sampled. An interrupt (IRQ) can also be set along with the EICUn_CNFGR: DATAVALID bit once 256 samples are stored in the SPLR registers. EICUn_CNFGR:OBSCH[4:0] shows the current interrupt pin being sampled. Writing '1' to EICUn_CNFGR:DATARESET clears the EICUn_CNFGR:DATAVALID bit and the interrupt.

Configuration Register (EICUn_CNFGR)

Figure 19-2. Configuration Register (EICUn_CNFGR)

EICUn_CNFGR																															
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**
read0	read0	read0	read0	read0	IRQEN	OBSEN	DATARESET	DATAVALID	BUSY	read0	OBSCH[4]	OBSCH[3]	OBSCH[2]	OBSCH[1]	OBSCH[0]	read0	read0	read0	read0	read0	read0	read0	read0	PRESCALE[5]	PRESCALE[4]	PRESCALE[3]	PRESCALE[2]	PRESCALE[1]	PRESCALE[0]	CLKSEL[1]	CLKSEL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	Rp0Wp1	Rp	Rp	Rp0	Rp	Rp	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-2. Configuration Register (EICUn_CNFGR) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26]	IRQEN	Interrupt Enable This bit enables the interrupt from the module. The interrupt is set when both the EICUn_CNFGR:DATAVALID and IRQEN bits are high. '0': Interrupt is disabled '1': Interrupt is enabled
[25]	OBSEN	Observation Enable This bit is used to enable the observation and sampling of the available 32 External Interrupt Pins. When the OBSEN bit is low, ongoing sampling is stopped immediately. With this bit going high, new sampling is started from 0-255. '0': Observation and sampling is disabled '1': Observation and sampling is enabled
[24]	DATARESET	Data Reset This bit is used to reset the EICUn_CNFGR:DATAVALID bit. '0': No effect '1': EICUn_CNFGR:DATAVALID bit is reset Reading this bit always returns '0'.
[23]	DATAVALID	Data Valid This bit is set when the data in the EICUn_SPLR is valid (256 samples are collected). '0': Data is not valid '1': Data is valid, As long as DATAVALID is '1', sampling of External Interrupt Pin (Pin INTk) is disabled (no further sample request is processed).
[22]	BUSY	Sampling Status This bit shows the sampling status of EICUn_CNFGR:OBSCH (Pin INTk). It is reset with EICUn_CNFGR:DATAVALID bit = '1' or EICUn_CNFGR:OBSEN bit = '0'. '0': Sampling of the observed channel is not ongoing '1': Sampling of observed channel is ongoing,
[21]	read0	-

Table 19-2. Configuration Register (EICUn_CNFGR) bits

Bit Position	Bit Field Name	Bit Description
[20:16]	OBSCH	Observed Channel These bits specify the External Interrupt Pin (Pin INTk) number being observed and sampled (out of the 32 available interrupt pins). Observed channel should be read only when EICUn_CNFGR:DATAVALID bit is '1'.
[15:8]	read0	-
[7:2]	PRESCALE	Prescale These bits decide the prescaler value for deriving sample clock from source clock. It can be programmed from 0 to 63. The sample clock calculation is shown in the 19.4 Notes on using the EICU. prescaler value = PRESCALE + 1 sample clock = source clock/ (prescaler value)
[1:0]	CLKSEL	Clock Select These bits select the source clock for sampling. '00': Slow RC clock (default) '01': RC clock '10': Main clock '11': Sub clock

19.2.2 External Interrupt Pin Enable Register (EICUn_IRENR)

The External Interrupt Pin Enable Register bits enable/disable the respective External Interrupt pins (Pin INTk) to be observed. In the event of simultaneous events in more than one enabled interrupt pin, priority is always given to the lower pin number, i.e. Pin INT0 has the highest priority and Pin INT31 has the lowest priority. All 256 samples are taken from one single interrupt pin. By default all the interrupt pins are disabled for observation.

External Interrupt Pin Enable Register (EICUn_IRENR)

Figure 19-3. External Interrupt Pin Enable Register (EICUn_IRENR)

EICUn_IRENR																															
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**
IREN[31]	IREN[30]	IREN[29]	IREN[28]	IREN[27]	IREN[26]	IREN[25]	IREN[24]	IREN[23]	IREN[22]	IREN[21]	IREN[20]	IREN[19]	IREN[18]	IREN[17]	IREN[16]	IREN[15]	IREN[14]	IREN[13]	IREN[12]	IREN[11]	IREN[10]	IREN[9]	IREN[8]	IREN[7]	IREN[6]	IREN[5]	IREN[4]	IREN[3]	IREN[2]	IREN[1]	IREN[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-3. External Interrupt Pin Enable Register (EICUn_IRENR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IREN	External Interrupt Pin Observe Enable These bits enable/disable the observation of the corresponding External Interrupt Pin. '0': Corresponding Pin INTk is disabled for observation and sampling '1': Corresponding Pin INTk is enabled for observation and sampling

19.2.3 Sample Registers 0~7 (EICUn_SPLR0~7)

These registers contain the sampled data of the interrupt pin (Pin INTk) captured. There are eight such 32-bit registers, storing a total of 256 samples. These registers act as a 256 bit shift register and the data in these registers are valid only when EICUn_CNFGR:DATAVALID bit is set (after collection of the 256 samples).

Only EICUn_SPLR0 is described here. Other registers (i.e EICUn_SPLR0, EICUn_SPLR1, EICUn_SPLR2, EICUn_SPLR3, EICUn_SPLR4, EICUn_SPLR5, EICUn_SPLR6, and EICUn_SPLR7) have similar bit fields.

Sample Register 0 (EICUn_SPLR0)

Figure 19-4. Sample Register 0 (EICUn_SPLR0)

EICUn_SPLR0																															
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**
SPL[31]	SPL[30]	SPL[29]	SPL[28]	SPL[27]	SPL[26]	SPL[25]	SPL[24]	SPL[23]	SPL[22]	SPL[21]	SPL[20]	SPL[19]	SPL[18]	SPL[17]	SPL[16]	SPL[15]	SPL[14]	SPL[13]	SPL[12]	SPL[11]	SPL[10]	SPL[9]	SPL[8]	SPL[7]	SPL[6]	SPL[5]	SPL[4]	SPL[3]	SPL[2]	SPL[1]	SPL[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-4. Sample Register 0 (EICUn_SPLR0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SPL	Sample 0 Register These bits store the level of the interrupt pin being sampled at isochronous steps.

Note: Sample 0 to Sample 7 registers store 256 samples of the levels of the enabled interrupt channel in isochronous steps. The first sample will be stored in EICUn_SPLR0[0] and the last sample will be stored in EICUn_SPLR7[31].

19.3 Operation of the EICU

The following section describes the operation of EICU in detail.

The following section explains the top level block diagram of EICU and its operation. The block diagram of the EICU is shown in Figure 19-1.

Priority encoder and channel select

EXT_IRk inputs come from the External Interrupts module and are the latch output of the first event in the corresponding Pin INTk. The channel select stores the channel (pin) number of the enabled channel (specified by EICUn_IRENR register) which had the first event. It is stored in EICUn_CNFGR:OBSCH[4:0]. In the event of a simultaneous event, the channel number is stored based on priority. Channel 0 has the highest priority and channel 31 has the lowest.

Clock request

With the first event on any of the enabled interrupt pins, Pin INTk (EXT_IRk = '1'), a clock request is sent to the clock controller block only if EICUn_CNFGR:CLKSEL[1:0] is programmed for the Slow or RC clock. If the EICUn_CNFGR:CLKSEL[1:0] is programmed for Main or Sub clock, it is assumed that the clocks are already available.

Note:

During power saving modes (which disables the Main and/or Sub clock), only the RC clock shall be used with the EICU.

Between the first event in any of the channels and the arrival of the selected clock (EICUn_CNFGR:CLKSEL[1:0] = Slow RC clock/RC clock), there can be a possible loss of levels on the interrupt pins (Pin INTk). The first activated interrupt pin (enabled pin) will be selected and sampled. For proper sampling, four times oversampling is suggested. This means, after recognizing an External Interrupt event, the RC clock must be started.

Clock settings

This block selects between the following four source clocks based on EICUn_CNFGR:CLKSEL[1:0] programming:

- Slow RC clock
- RC clock
- Main clock and
- Sub clock

Prescaler bits scale the selected source clock based on EICUn_CNFGR:PRESCALE[5:0] programming and generate a sample clock.

Signal sampling and storing

This module captures 256 samples of the selected channel (Pin INTk) based on the channel select. The sampling is done with respect to the sample clock and the data is stored in eight 32-bit Sample Registers (SPLR). During the sampling period, the EICUn_CNFGR:BUSY bit is set and it is reset with EICUn_CNFGR:DATAVALID = '1'. The EICUn_CNFGR:DATAVALID bit is set when 256 samples are collected in the SPLR. When EICUn_CNFGR:DATAVALID = '1', the interrupt (IRQ) is also sent to the host in case EICUn_CNFGR:IRQEN is set.

Enabling/disabling of the capture function

Using the EICUn_CNFGR:OBSEN bit, the sampling function of the External Interrupt Pins can be enabled or disabled. If EICUn_CNFGR:OBSEN = '1', sampling starts with an event on any one of the enabled External Interrupt channels.

If EICUn_CNFGR:OBSEN = '0', the SPLR are flushed, and the EICUn_CNFGR:DATAVALID and EICUn_CNFGR:BUSY bits are reset.

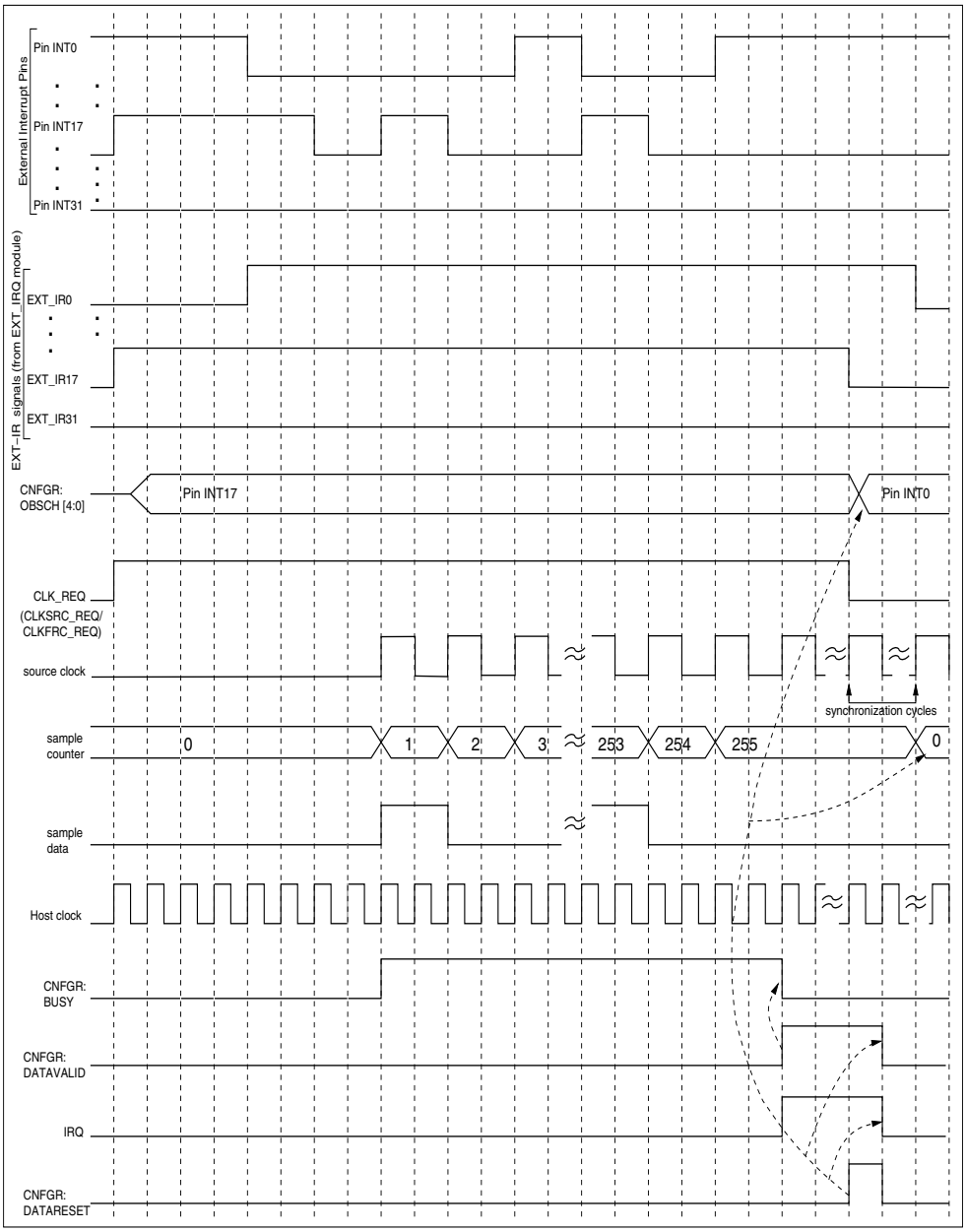
Example operation of EICU

The timing diagram in Figure 19-5 typically shows the operation of the EICU. Here the assumption is that all the External Interrupt Pins are enabled for event monitoring and EICUn_CNFGR:OBSEN is set for the entire period. Pin INT17 has the first event followed by Pin INT0. All other interrupt pins are inactive. With the event on Pin INT17, a clock request is generated (EICUn_CNFGR:CLKSEL[1:0] = Slow RC/RC clock) and channel select is updated to Pin INT17. Even in the event of another

event on Pin INT0, channel select is not updated with Pin INT0. EICUn_CNFGR:BUSY is set when sampling starts and is reset once EICUn_CNFGR:DATAVALID is set. EICUn_CNFGR:DATAVALID is set after 256 samples have been collected and stored in the Sample Registers. An interrupt is sent to the host once EICUn_CNFGR:DATAVALID is set (assuming EICUn_CNFGR:IRQEN = '1') and it is de-asserted with EICUn_CNFGR:DATAVALID = '0'. With EICUn_CNFGR:DATARESET = '1', EICUn_CNFGR:DATAVALID is de-asserted.

Note: External Interrupt generation and clear logic is independent of the EICU.

Figure 19-5. Example operation of EICU

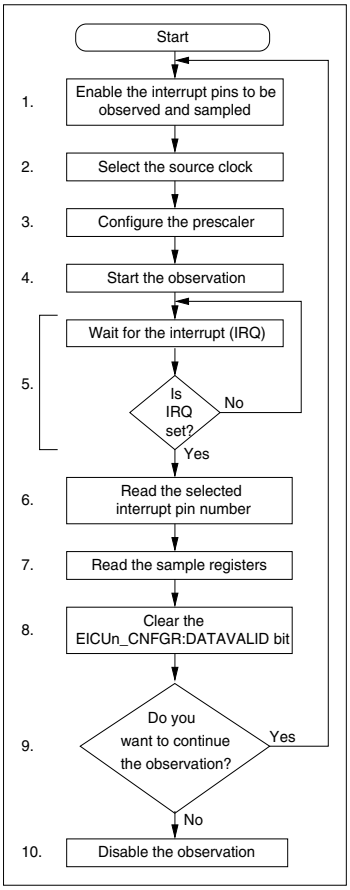


19.4 Notes on using the EICU

This section describes how to use the EICU.

Steps in programming the EICU

Figure 19-6. Programming the EICU



General steps a programmer shall follow while using the EICU:

1. After the system reset, set the corresponding bits in EICUn_IRENR register for the interrupt pins to be observed and sampled. By default all the interrupt pins are disabled for observation. To sample any interrupt pin, the corresponding enable for that interrupt should also be enabled in the External Interrupt block. Event detection (signal detection) on the interrupt pin depends on the External Interrupt block settings (falling/rising edge, high/low level). For more details, refer to 24. External Interrupts.
2. Program the EICUn_CNFGR:CLKSEL[1:0] bits to select the desired source clock from Slow RC clock, RC clock, Main clock, and Sub clock. Note that at reset the Slow RC clock is selected, but by default Slow RC Osc is disabled. Also requirement of glitch free clock switching is that both (old and new) clock sources to be enabled at the time of EICUn_CNFGR:CLKSEL[1:0] change. So, before switching from Slow RC Osc clock to Main clock, the user needs to ensure that both oscillators are enabled. Once switching is done, the old clock (Slow RC Osc) can be disabled. However, the Main Osc still needs to be enabled during Power Saving State (PSS) so that it can sample the data at wakeup.
3. Set the EICUn_CNFGR:PRESCALE[5:0] bits to select the appropriate scaling factor for the source clock.
   a. Sampling frequency = source frequency/(EICUn_CNFGR:PRESCALE[5:0] +1)
   b. Sampling frequency should be preferably four times faster than the fastest baud rate available amongst the enabled External Interrupt Pins. Source clock and prescaler values should be programmed accordingly.
4. Set the EICUn_CNFGR:OBSEN to start the observation.

5. With the first event in any of the enabled interrupt pins (EXT_IRk = '1'), the clock is requested (CLKSRC_REQ/CLKFRC_REQ), if the EICUn_CNFGR:CLKSEL[1:0] is programmed for Slow RC clock or RC clock. After the arrival of the source clock (refer to the datasheet for the time required for the Slow RC and RC clock), the sampling frequency is calculated based on the prescaler setting. The External Interrupt Pin number with the first event is stored in EICUn_CNFGR:OBSCH[4:0] (channel select). The Pin INTk corresponding to the selected channel is sampled for 256 samples at the sampling frequency and the samples are stored in the SPLR. After capturing 256 samples, EICUn_CNFGR:DATAVALID bit is set. Simultaneously the interrupt (IRQ) is also set if EICUn_CNFGR:IRQEN = '1'. The user should wait for EICUn_CNFGR:DATAVALID = '1' to read the SPLR. The clock request is de-asserted after the EICUn_CNFGR:DATAVALID bit is set.
6. The selected channel number can be read from EICUn_CNFGR:OBSCH[4:0]. In the event of simultaneous events in more than one enabled External Interrupt Pin, priority is always given to the lower channel number. On other words Pin INT0 has the highest priority and Pin INT31 has the lowest priority.
7. After the arrival of the interrupt, the user can read the data from the 32-bit EICUn_SPLR0~7 registers.
8. The user should set EICUn_CNFGR:DATARESET to clear EICUn_CNFGR:DATAVALID and subsequently the interrupt (IRQ) after reading the EICUn_SPLR0~7 registers. To start a new observation, EICUn_CNFGR:DATAVALID should be always cleared. EICUn_CNFGR:DATAVALID is cleared only when the user sets the EICUn_CNFGR:DATARESET bit or EICUn_CNFGR:OBSEN is cleared

   Note:

   tAfter writing EICUn_CNFGR:DATARESET, a few cycles are needed to execute because of the synchronization processes. Thus, a waiting period of at least 2 cycles of CLK_CFG_PD1 and 2 cycles of the used sampling clock are required before the EICU is ready for a new recording.
9. To continue the observation with the same set of enabled Pin INTk channels, the source clock and prescaler settings, go directly to step 5.
10. If you want to change any programmed values or want to discontinue observation, disable the EICUn_CNFGR:OBSEN bit. The observation and sampling process is inactive when the EICUn_CNFGR:OBSEN bit is '0' or EICUn_CNFGR:DATAVALID is '1'

Example usage of EICU for CAN

In the event that wake-up via CAN is required and the wake-up message needs to be detected, the EICU can be used. Based on the assumption that the MCU is in low-power mode, the steps below are needed to configure the EICU for wake-up using CAN and must be carried out before the MCU goes to low-power mode:

1. Enable the CAN Rx interrupt in the External Interrupt block for interrupt generation according to the CAN bus dominant bit setting.
2. Enable observation of the interrupt pin (EICUn_IRENR) connected to the CAN RX pin.
3. Select the appropriate capture frequency depending on CAN bit rate.: For example, if CAN is working at 1 Mbps, the sampling frequency should be configured for 4 MHz.
4. Enable observation by setting the EICUn_CNFGR:OBSEN bit.

Operation of the EICU

1. The device enters low-power state.
2. In the event that a wake-up message is received, the External Interrupt block will detect an event on the CAN RX interrupt pin.
3. This triggers a wake up sequence of the MCU and EICU.
4. After the RC clock is operable (refer to the datasheet for time required for Slow RC and RC clock), capturing the CAN bitstream starts.
5. The interrupt pin is sampled and the 256 samples are stored in the SPLR (EICUn_SPLR0~7).
6. After the 256 samples have been collected, the EICUn_CNFGR:DATAVALID bit will be set and an interrupt is generated if enabled.
7. After the device and application has finished the wake-up procedure, the application can check the capture values and recalculate the received message.

20. Timing Protection Unit

This chapter explains the functions and operations of the Timing Protection Unit (TPU).

20.1 Outline of TPU

For safe and accurate timing protection of the system it is necessary for the operating system to control the execution time, lock time, inter arrival time, and deadline time for various tasks. The motivation behind the TPU module is to implement a flexible hardware that supports timing protection function of the operating system. This section describes the features and the block diagram of the TPU in detail.

Features of TPU

Modern operating systems (especially time triggered systems) require more means to supervise the timing behavior of single task. A single task might have:

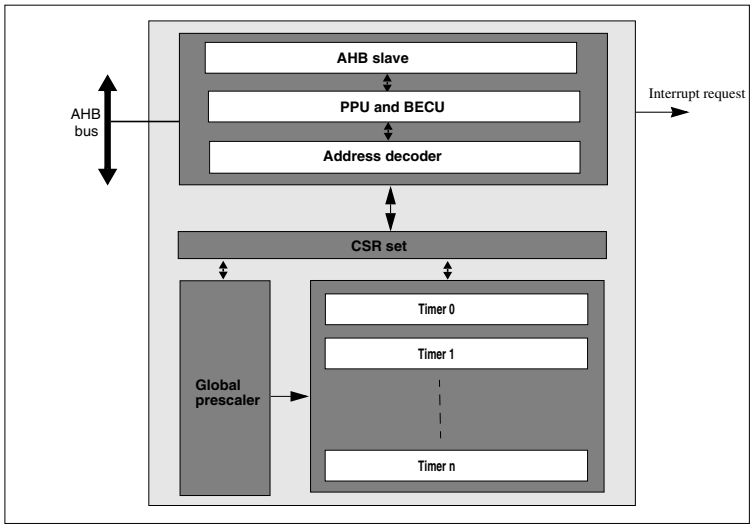
- A maximum overall runtime
- A deadline when it must be finished after its start
- A maximum time it is allowed to block interrupts of a certain level
- A maximum time it is allowed to disable interrupts globally
- A minimum rate of re-activation

To meet this requirement the FCR4 Cluster series provides a hardware support using the Timing Protection Unit having the following features:

- Maximum up to eight identical timers which can be used for execution time protection, locking time protection, inter-arrival time protection, or deadline protection
- 24-bit up-counters with programmable normal and overflow mode
- Programmable end-count for normal mode. Interrupt is generated when count is greater than or equal to the end-count value and then timer resets to '0'
- NMI interrupt is generated on counter overflow mode
- Automatic restart of timer when Free Running Timer bit is set
- Global linear prescaler (1 to 64) to scale down the system clock frequency to required frequency, as input to individual timer prescaler
- Individual timer prescaler to support four different software-programmable frequencies (Fin, Fin/2, Fin/4, Fin/16)
- Current counter value readable by software
- Start, stop, and continue options for each timer by software
- Provides status information of each timer (stopped or active)
- Register modification allowed only in privileged mode
- 64-bit bus interface for accessing TPU internal registers
- Supports debug mode for stalling the timers

Block diagram of TPU

Figure 20-1. Block diagram of TPU



- TPU register set

  The operation of TPU can be controlled and monitored through its Configuration and Status Register (CSR) set.
- Global prescaler

  Global prescaler is implemented to scale down the input AHB clock frequency in linear range of 1 to 64.
- Timers

  Each timer has its individual timer prescaler and timer logic.

  Timer prescaler is implemented to scale down the input frequency from global prescaler to following range of frequencies: divide by 1, divide by 2, divide by 4, and divide by 16. Timer logic implements the 24-bit up counter with start, stop, and continue logic.
- Peripheral protection and Bus Error Collection

Peripheral protection logic in TPU protects the TPU CSRs from the illegal AHB accesses, based on the access levels defined for individual registers. Occurrence of faults are signalled by TPU to the BECU, so that the BECU can collect the information about the transfer.

20.2 TPU registers

This section describes the registers of the TPU in detail.

Registers of TPU

The following registers are available for each instance of TPU:

- TPU Unlock Register (TPUn_UNLOCK)
- TPU Lock Status Register (TPUn_LST)
- TPU Configuration Register (TPUn_CFG)
- TPU Timer Interrupt Request Register (TPUn_TIR)
- TPU Timer Status Register (TPUn_TST)
- TPU Timer Interrupt Enable Register (TPUn_TIE)
- TPU Module ID Register (TPUn_MID)
- TPU Timer Control Register 0 (TPUn_TCN00~07)
- TPU Timer Control Register 1 (TPUn_TCN10~17)
- TPU Timer Current Count Register (TPUn_TCC0)

Programmer's model

Each instance of TPU module uses 1 KB of address space for mapping its own Control and Status Registers (CSRs). All registers of TPU are shown in figure below. The registers can be accessed with 8-bit, 16-bit, 32-bit, and 64-bit accesses.

Table 20-1. Memory layout of TPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	TPUn_LST 00000000 00000000 00000000 00000001				TPUn_UNLOCK 00000000 00000000 00000000 00000000			
0x00000008	TPUn_TIR 00000000 00000000 00000000 00000000				TPUn_CFG 00000000 00000000 00000000 00000000			
0x00000010	TPUn_TIE 00000000 00000000 00000000 00000000				TPUn_TST 00000000 00000000 00000000 00000000			
0x00000018	read0 00000000 00000000 00000000 00000000				TPUn_MID 00000000 00000000 00000000 00000000			
0x00000020 - 0x00000028	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000030	TPUn_TCN01 00000000 00000000 00000000 00000000				TPUn_TCN00 00000000 00000000 00000000 00000000			
0x00000038	TPUn_TCN03 00000000 00000000 00000000 00000000				TPUn_TCN02 00000000 00000000 00000000 00000000			
0x00000040	TPUn_TCN05 00000000 00000000 00000000 00000000				TPUn_TCN04 00000000 00000000 00000000 00000000			
0x00000048	TPUn_TCN07 00000000 00000000 00000000 00000000				TPUn_TCN06 00000000 00000000 00000000 00000000			
0x00000050	TPUn_TCN11 00000000 00000000 00000000 00000000				TPUn_TCN10 00000000 00000000 00000000 00000000			

Table 20-1. Memory layout of TPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000058	TPUn_TCN13 00000000 00000000 00000000 00000000				TPUn_TCN12 00000000 00000000 00000000 00000000			
0x00000060	TPUn_TCN15 00000000 00000000 00000000 00000000				TPUn_TCN14 00000000 00000000 00000000 00000000			
0x00000068	TPUn_TCN17 00000000 00000000 00000000 00000000				TPUn_TCN16 00000000 00000000 00000000 00000000			
0x00000070	TPUn_TCC1 00000000 00000000 00000000 00000000				TPUn_TCC0 00000000 00000000 00000000 00000000			
0x00000078	TPUn_TCC3 00000000 00000000 00000000 00000000				TPUn_TCC2 00000000 00000000 00000000 00000000			
0x00000080	TPUn_TCC5 00000000 00000000 00000000 00000000				TPUn_TCC4 00000000 00000000 00000000 00000000			
0x00000088	TPUn_TCC7 00000000 00000000 00000000 00000000				TPUn_TCC6 00000000 00000000 00000000 00000000			

Notes:

The suffix 'n' denotes the instance number of the TPU.

While writing, all 'reserved' field must be reset to '0'. While reading, all 'reserved' field must be ignored.

20.2.1 TPU Unlock Register (TPUn_UNLOCK)

This register unlocks the various configuration registers. This protects them from being written accidently.

TPU Unlock Register (TPUn_UNLOCK)

Figure 20-2. TPU Unlock Register (TPUn_UNLOCK)

TPUn_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-2. TPU Unlock Register (TPUn_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	TPU Unlock This register protects the TPU module from being modified accidently by software. The static configuration registers cannot be written until this register has been written with a specific correct value. The correct value for unlocking (0xACC5A110) can be written only in privileged mode. The read to this register always returns '0'. To lock the TPU again, software must write another value specific to lock (0xB10CACC5). Illegal access to static configuration registers or writing value other than lock or unlock value to this register causes protection error. The list of registers in TPU which are protected by this register are: *TPUn_CFG. *TPUn_TCN10~TPUn_TCN17.

Note: TPUn_UNLOCK register requires entire 32-bit value to make decision on lock status. Hence it shall be written only by using 32- or 64-bit accesses. It shall not be written by using 8- or 16-bit accesses.

20.2.2 TPU Lock Status Register (TPUn_LST)

This is a read-only register that indicates the locked or unlocked status of the TPU.

TPU Lock Status Register (TPUn_LST)

Figure 20-3. TPU Lock Status Register (TPUn_LST)

TPUn_LST																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	LST
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 20-3. TPU Lock Status Register (TPUn_LST) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	LST	TPU Lock Status This bit provides locked/unlocked status of TPU. '0': TPU is unlocked '1': TPU is locked

20.2.3 TPU Configuration Register (TPUn_CFG)

TPU Configuration Register is used to configure global prescaler of TPU and for setting the global interrupt enable bit. It is also used for enabling/disabling the debug mode.

TPU Configuration Register (TPUn_CFG)

Figure 20-4. TPU Configuration Register (TPUn_CFG)

TPUn_CFG																															
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**
read0	read0	read0	read0	read0	read0	read0	DBGE	GLBPSE	read0	GLBPS[5]	GLBPS[4]	GLBPS[3]	GLBPS[2]	GLBPS[1]	GLBPS[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	INTE
R0	R0	R0	R0	R0	R0	R0	RWP	RWP	R0	RWP	RWP	RWP	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-4. TPU Configuration Register (TPUn_CFG) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	DBGE	Debug Mode Enable '0': Debug mode is disabled '1': Debug mode is enabled When DBGE is set to '1' and the processor is in debug state, all timer counters are stalled at their current state. However, this doesn't affect the Timer Status bits TPUn_TST:TS[7:0]. For more details on debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual. Refer to Section 18.3 for more information on debug mode.
[23]	GLBPSE	Global Prescaler Enable '0': Global prescaler is disabled, global prescaler counter will stop after next increment pulse. As a result all the timers will also stop. '1': Global prescaler is enabled, global prescaler counter is running
[22]	read0	-

Table 20-4. TPU Configuration Register (TPUn_CFG) bits

Bit Position	Bit Field Name	Bit Description
[21:16]	GLBPS	Global Prescaler The requirement for a TPU is to have a scaled down frequency for counting from the system clock. These global prescaler (GLBPS[5:0]) register bits provide the feature to prescale the system clock to a lower frequency for use within the TPU. These bits provide a prescaler common to all timers, or in other words the clock output from this prescaler feeds all the different timing protection timers implemented in the TPU module specific to the device. The prescaler values shown below divide the input system clock to lower frequencies. The bits indicate an absolute division and are not encoded. '000000': Divide by 1 '000001': Divide by 2 '000010': Divide by 3 ... ... '111111': Divide by 64 **Note:** Refer to the device specific datasheet for the number of timers implemented in the device.
[15:8]	read0	-
[7:1]	read0	-
[0]	INTE	TPU Interrupt Enable This bit provides a global interrupt disable feature. The pending interrupts still get flagged in the TPUn_TIR. '0': Interrupt generation is disabled. Interrupt is not generated to CPU independent of status of TPUn_TIE:IE[m] and TPUn_TIR:IR[m] bits '1': Interrupt generation is enabled. Interrupt is generated to CPU if TPUn_TIE:IE[m] = '1' and TPUn_TIR:IR[m] = '1'

20.2.4 TPU Timer Interrupt Request Register (TPUn_TIR)

TPU Timer Interrupt Request Register provides interrupt status of each timer.

TPU Timer Interrupt Request Register (TPUn_TIR)

Figure 20-5. TPU Timer Interrupt Request Register (TPUn_TIR)

TPUn_TIR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IR[7]	IR[6]	IR[5]	IR[4]	IR[3]	IR[2]	IR[1]	IR[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-5. TPU Timer Interrupt Request Register (TPUn_TIR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	IR	Interrupt Request These bits indicate whether a particular timer has raised an interrupt flag or if there is a pending interrupt from timer. Each bit corresponds to a particular timer. IR[m] = '0':No interrupt request from the timer-n IR[m] = '1':Interrupt flag is set or there is a pending interrupt request from the timer-n. **Notes:** 1. **m** in IR[m] indicates timer number. 2. Interrupt to the CPU is generated when IR[m] = '1' and TPUn_TIE:IE[m] = '1' and TPUn_CFG:INTE = '1'. 3. Refer to the device specific datasheet for the number of timers implemented in the device.

20.2.5 TPU Timer Status Register (TPUn_TST)

This register provides active status (running or stopped) of each timer.

TPU Timer Status Register (TPUn_TST)

Figure 20-6. TPU Timer Status Register (TPUn_TST)

TPUn_TST																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TS[7]	TS[6]	TS[5]	TS[4]	TS[3]	TS[2]	TS[1]	TS[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-6. TPU Timer Status Register (TPUn_TST) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	TS	Timer Status These bits indicate the active status of each timer in the TPU module. It indicates which of the timers are currently running and which are stopped. TS[m] = '0':Timer-m is not active or stopped TS[m] = '1':Timer-m is active and running **Notes:** 1. m in TS[m] indicates timer number. 2. Refer to the device specific datasheet for the number of timers implemented in the device.

20.2.6 TPU Timer Interrupt Enable Register (TPUn_TIE)

This register enables or disables each timer interrupt.

TPU Timer Interrupt Enable Register (TPUn_TIE)

Figure 20-7. TPU Timer Interrupt Enable Register (TPUn_TIE)

TPUn_TIE																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IE[7]	IE[6]	IE[5]	IE[4]	IE[3]	IE[2]	IE[1]	IE[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-7. TPU Timer Interrupt Enable Register (TPUn_TIE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:0]	IE	Interrupt Enable These bits indicate whether the interrupt generated by a particular timer is signaled to the CPU or not. IE[m] = '0':Interrupt is disabled IE[m] = '1':Interrupt is enabled **Notes:** 1. **m** in IE[m] indicates timer number. 2. Interrupt to the CPU is generated when TPUn_TIR:IR[m] = '1' and IE[m] = '1' and TPUn_CFG:INTE = '1'. 3. Refer to the device specific datasheet for the number of timers implemented in the device.

20.2.7 TPU Module ID Register (TPUn_MID)

This is a read-only register with a unique module identification number, which identifies the version of the TPU module used in the MCU.

TPU Module ID Register (TPUn_MID)

Figure 20-8. TPU Module ID Register (TPUn_MID)

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

Table 20-8. TPU Module ID Register (TPUn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	TPU Module ID The TPU Module ID (TPUn_MID) implemented in the device may vary from device to device. This register identifies the particular version of the hardware used in the device. This register helps in developing software specific to the hardware version implemented in the device. **Note:** Refer to the device specific datasheet for version number details.

20.2.8 TPU Timer Control Register 0 (TPUn_TCN00~07)

This is control register for timer. This register configures the end count or preload value for timer. It also provides bits for starting, stopping and continuing of the timer. This register also provides bits for clearing the interrupt flag and also bits for setting and clearing of interrupt enable bit. This register is repeated for each instance of timer (TPUn_TCN00~07).

TPU Timer Control Register 0 (TPUn_TCN00~07)

Figure 20-9. TPU Timer Control Register 0 (TPUn_TCN00)

TPUn_TCN00																															
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**
START	STOP	CONT	IES	IEC	IRC	read0	read0	ECPL[23]	ECPL[22]	ECPL[21]	ECPL[20]	ECPL[19]	ECPL[18]	ECPL[17]	ECPL[16]	ECPL[15]	ECPL[14]	ECPL[13]	ECPL[12]	ECPL[11]	ECPL[10]	ECPL[9]	ECPL[8]	ECPL[7]	ECPL[6]	ECPL[5]	ECPL[4]	ECPL[3]	ECPL[2]	ECPL[1]	ECPL[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-9. TPU Timer Control Register 0 (TPUn_TCN00) bits

Bit Position	Bit Field Name	Bit Description
[31]	START	Start '0': No effect '1': Starts the timer from '0' and will also load the new end value specified in the TPUn_TCN00:ECPL bits Reading this bit always returns '0'.
[30]	STOP	Stop '0': No effect '1': Stops the timer, but does not clear it Reading this bit always returns '0'.
[29]	CONT	Continue '0': No effect, '1': Starts the timer from a previously stalled value and will also load the new end value from the ECPL bits Reading this bit always returns '0'. TPUn_TCN00:START, TPUn_TCN00:STOP and CONT bits shall not be set simultaneously.

Table 20-9. TPU Timer Control Register 0 (TPUn_TCN00) bits

Bit Position	Bit Field Name	Bit Description
[28]	IES	Interrupt Enable Set This bit is used to set timer interrupt enable bit. '0': No effect, '1': Sets the interrupt enable bit (TPUn_TIE:IE[m]) Reading this bit always returns '0'. **Notes:** 1. m in TPUn_TIE:IE[m] indicates timer number. 2. If set at the same time as TPUn_TCN00:IEC, then this bit takes priority.
[27]	IEC	Interrupt Enable Clear This bit is used to clear timer interrupt enable bit. '0': No effect, '1': Clears the interrupt enable bit (TPUn_TIE:IE[m]) Reading this bit always returns '0'. **Notes:** 1. m in TPUn_TIE:IE[m] indicates timer number. 2. If this bit is set at the same time as TPUn_TCN00:IES, then TPUn_TCN00:IES takes priority.
[26]	IRC	Interrupt Request Clear This bit is used to clear timer interrupt flag. '0': No effect, '1': Clears the interrupt flag in TPUn_TIR:IR[m] Reading this bit always returns '0'. **Note:** m in TPUn_TIR:IR[m] indicates timer number.
[25:24]	read0	-
[23:0]	ECPL	End Count or Pre Load When the timers in the TPU module are programmed for normal mode of operation, internally these timers run as up-counters. These register bits defined in the end counter preload indicate the end count value. When the same timers are programmed for free running mode of operation, the ECPL bits indicate the preload value for these timers. The preload value can be used by software to program a value which can cause the timer to run to the maximum limit of 0xFFFFFF faster for testing purpose. The mode of the timer is programmed in the TPUn_TCN10:TMOD bit. **Note:** Refer to the device specific datasheet for the number of timer implemented in the device.

Note: The bit description for registers TPUn_TCN01~07 is same as for the TPUn_TCN00 register.

20.2.9 TPU Timer Control Register 1 (TPUn_TCN10~17)

This register configures the individual prescalers for each timer. It also provides bits for changing the mode of the timer and preload enable bits. This register is repeated for each instance of timer (TPUn_TCN10~TPUn_TCN17).

TPU Timer Control Register 1 (TPUn_TCN10)

Figure 20-10. TPU Timer Control Register 1 (TPUn_TCN10)

TPUn_TCN10																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PL	FRT	TMOD	PS[1]	PS[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-10. TPU Timer Control Register 1 (TPUn_TCN10) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4]	PL	Preload This bit is used to configure the timer to start counting with pre-defined value instead of '0'. This bit is applicable only when timer is in overflow mode. '0': Preload mode is not active '1': Preload mode is active
[3]	FRT	FRT Free Running Timer This bit is used to configure the timer in free-running mode. '0': Free running mode is not active '1': Free running mode is active
[2]	TMOD	TPU Mode This bit is used to configure the timer mode of operation. '0': Normal mode '1': Overflow mode

Table 20-10. TPU Timer Control Register 1 (TPUn_TCN10) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	PS	Individual Prescaler These bits provide an option to scale the clock generated from the global prescaler to the following frequencies: '00': Divide by 1 '01': Divide by 2 '10': Divide by 4 '11': Divide by 16 **Note:** Refer to the device specific datasheet for the number of timers implemented in the device.

Note:

The bit description for registers TPUn_TCN11~TPUn_TCN17 is same as TPUn_TCN10 register.

The following table explains the various mode of operation of timer with reference to its configuration bits.

Table 20-11. Timer modes of operation

TMOD	FRT	PL	Timer modes of operation
'0'	'0'	'0'	Timer in normal mode
'0'	'0'	'1'	Timer in normal mode
'0'	'1'	'0'	Timer in normal mode but will restart automatically from '0'
'0'	'1'	'1'	Timer in normal mode but will restart automatically from '0'
'1'	'0'	'0'	Timer in overflow mode
'1'	'0'	'1'	Timer in overflow mode, on start will start with preload value
'1'	'1'	'0'	Timer in overflow mode but will restart automatically from '0'
'1'	'1'	'1'	Timer in overflow mode but will restart automatically from preload value

20.2.10 TPU Timer Current Count Register (TPUn_TCC0)

This is a read-only register which reads the current count value for each of the timers. This register is repeated for each instance of timer (TPUn_TCC0 to TPUn_TCC7).

TPU Timer Current Count Register (TPUn_TCC0)

Figure 20-11. TPU Timer Current Count Register (TPUn_TCC0)

TPUn_TCC0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	TCC[23]	TCC[22]	TCC[21]	TCC[20]	TCC[19]	TCC[18]	TCC[17]	TCC[16]	TCC[15]	TCC[14]	TCC[13]	TCC[12]	TCC[11]	TCC[10]	TCC[9]	TCC[8]	TCC[7]	TCC[6]	TCC[5]	TCC[4]	TCC[3]	TCC[2]	TCC[1]	TCC[0]
R0	R0	R0	R0	R0	R0	R0	R0	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-12. TPU Timer Current Count Register (TPUn_TCC0) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:0]	TCC	Timer Current Count These bits indicate the current timer value. The software can read this register at any point of time to check the current value of a particular timer. It should however be noted that the timer is not stopped on the read by CPU. Hence the timer will continue even after a particular read has occurred. The actual value recognized by the CPU after the bus latency may be different from the exact value of the timer at that instant. **Note:** Refer to the device specific datasheet for the number of timers implemented in the device.

Note: The bit description for registers TPUn_TCC1 to TPUn_TCC7 is same as TPUn_TCC0 register.

20.3 Operation of TPU

Each of the TPU timer can be configured in following different modes of operation.

Normal mode

In normal mode, the timer works like a normal up counter. The up count value is compared with the end count value programmed in the timer register (TPUn_TCN00:ECPL[23:0] bits). An interrupt flag is set if counter value reaches a value equal to or greater than the end value. Interrupt request is generated if interrupt flag is set and if corresponding interrupt is also enabled.Timer can be started by writing '1' to TPUn_TCN00:START bit. On start, timer always starts from '0'.

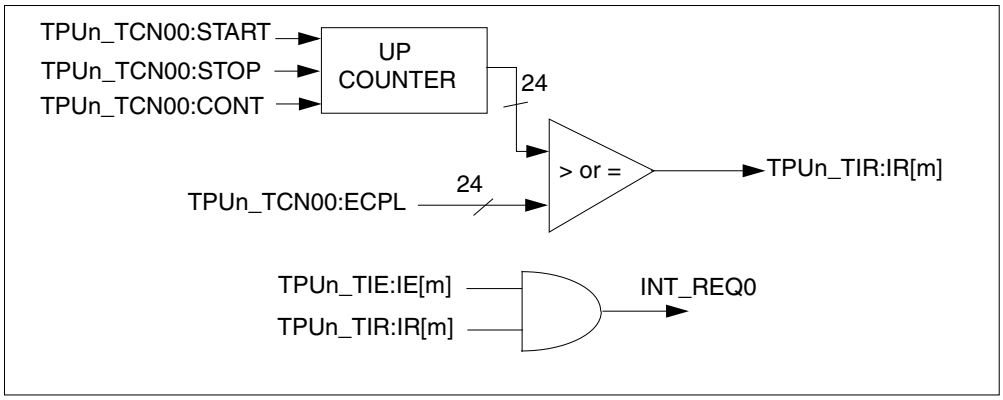
Timer status is active (TPUn_TST:TS = '1') when timer is running. Timer status changes to not active (TPUn_TST:TS = '0') when interrupt flag (TPUn_TIR:IRn) is set.

Alternately timer can be stopped by writing '1' to timer stop (TPUn_TCN00:STOP) bit on which timer status changes to stopped (TPUn_TST:TS = '0').

Note: On stop, timer value is stalled to current value and not set to '0'.

Timer can also be started by writing '1' to TPUn_TCN00:CONT bit. On continue, timer starts from previously stalled value and timer status is also changed to active (TPUn_TST:TS = '1').

Figure 20-12. Timer in normal mode (TPUn_TCN10:TMOD = ’0’)



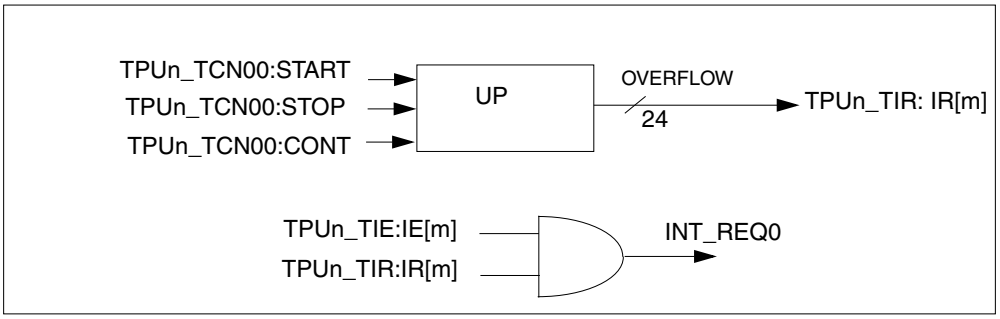
Overflow mode

In overflow mode, interrupt flag is set on counter overflow (i.e. on timer increment pulse and counter value is 0xFFFFFF). The counter stops counting after overflow event.

TPU also provides configuration to preload the timer with a preprogrammed value. This can be done by programming the preload value in TPUn_TCN00:ECPL and setting TPUn_TCN10:PL bit. On start, timer starts from '0' (when TPUn_TCN10:PL = '0') or preload value (when TPUn_TCN10:PL = '1').

This operates in the same way as in normal mode for TPUn_TCN00:STOP and TPUn_TCN00:CONT bits.

Figure 20-13. Timer in overflow mode (TPUn_TCN10:TMOD = ’1’)



Free-running mode

TPU also provides a configuration bit for free-running bit (TPUn_TCN10:FRT) mode of timer. This bit is applicable for both above-mentioned modes (normal and overflow).

If this bit is set, then timer restarts automatically after interrupt flag is set.

In normal mode, timer restarts from '0'.

In overflow mode, timer restarts from '0' (when TPUn_TCN10:PL = '0') or preload value (when TPUn_TCN10:PL = '1').

As timer keeps running even after interrupt flag is set, timer status also remains active (TPUn_TST:TS = '1'). Interrupt flag is set again (or remains set if not cleared before) every time interrupt event (as explained above in normal and overflow mode) occurs.

Debug mode

When the TPUn_CFG:DBGE is set to '1' and the processor is in debug state, all timers are stalled to their current state but timer status is not changed.

As soon as this condition is released, timers continue running from their current states.

For the definition of debug state, refer to Section 11.8 of the Arm Cortex-R4 Technical Reference Manual.

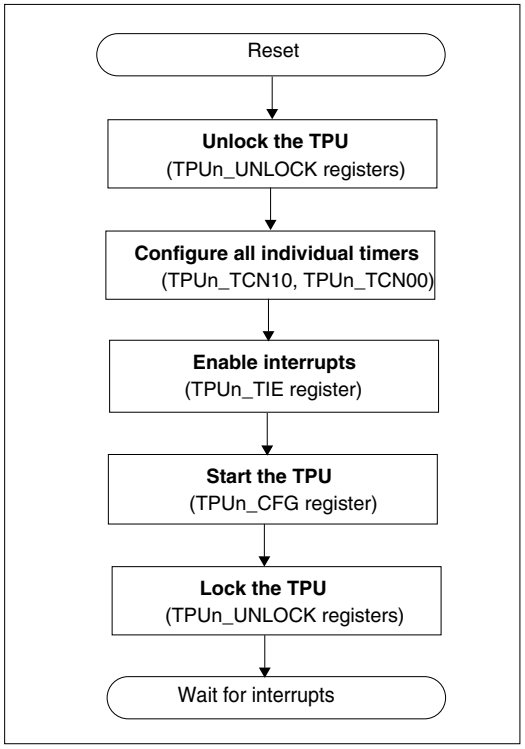
20.4 TPU flowchart

Following section with the help of flowcharts provides information on how TPU is programmed using its CSRs.

Starting all timers at once

Figure 20-14 shows in brief, the procedure for configuring the TPU to start all the timer at once.

Figure 20-14. Starting all timers at once



Flowchart for starting any one timer

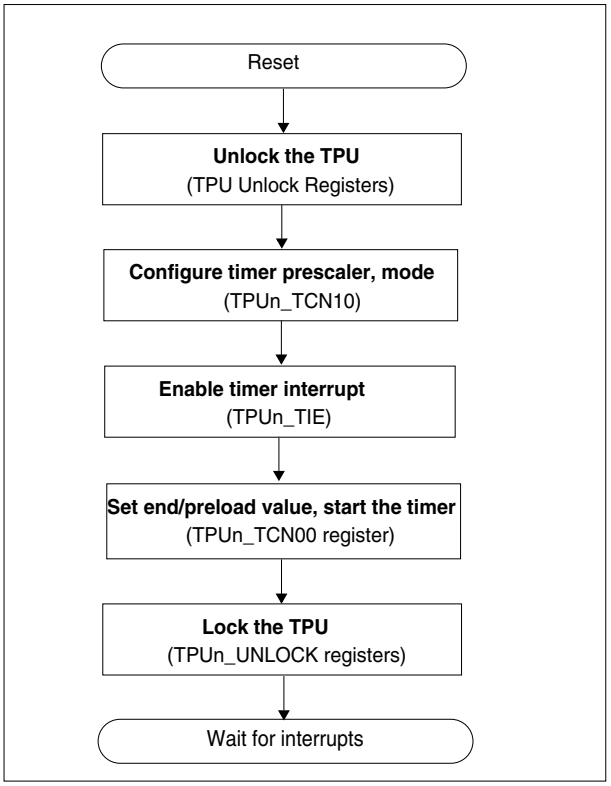
Figure 20-15 shows in brief the procedure for configuring the TPU to start any timer at any time. Assumption is that some other timers are already running.

Note:

Since global prescaler is already active in this case, actual time taken by timer to generate an interrupt can be one global prescaler inaccurate (i.e. max one global prescaler less).

e.g. Assuming TPUn_CFG:GLBPS[5:0] = 12, global prescaler is already running. If software starts any timer at a particular clock instance and increment pulse from global prescaler is generated immediately after two clocks, actual time taken by timer to generate interrupt would be 10 clocks less.

Figure 20-15. Starting any one timer



21. Peripheral Protection Unit

This chapter explains the functions and operations of the Peripheral Protection Unit (PPU).

21.1 Outline of Peripheral Protection Unit

This section describes the features and the block diagram of the Peripheral Protection Unit.

Features of Peripheral Protection Unit

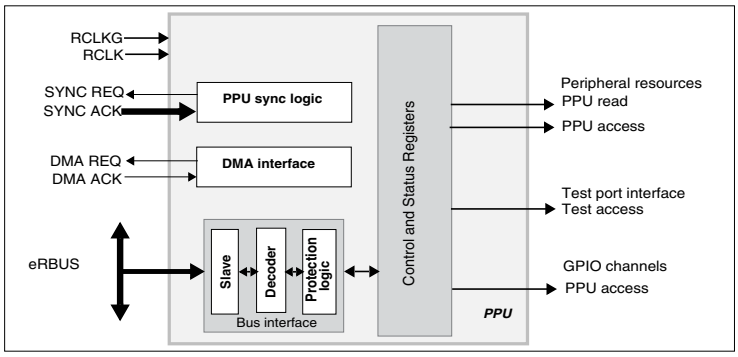
The PPU module provides protection attributes to various peripheral resources (hereafter referred to as peripherals) and GPIO channels. It controls the read and access protection attributes for peripherals and access protection attributes for GPIO channels. The PPU can only block non-privileged access.

- Provides read and access protection attribute for peripherals. For more details on PPU attributes refer to Table 21-14
- Supports access protection attribute for 512 GPIO channels
- Supports read and access attribute for upto 512 peripherals
- Provides self-synchronization mechanism to support peripherals on different clock domains
- Provides protection for all test registers on the device
- Supports DMA access for register configuration

Block diagram of Peripheral Protection Unit

The Figure 21-1 shows the internal block diagram of the PPU module.

Figure 21-1. Block diagram of Peripheral Protection Unit



Control and Status Registers

The operation of PPU can be controlled and monitored through Control and Status Registers (CSR). For details of the CSR refer to 21.2 PPU registers.

PPU sync logic

The PPU provides protection attributes for various peripherals. These attributes are used by the peripherals to protect against illegal access of its register space by the software. The PPU needs to ensure that the attributes are stable in the respective peripherals, before the software can access the peripheral register space. The software needs to access the peripheral which

are on different clock domains only after the protection attributes from PPU are stable. The PPU sync logic guarantees synchronization of PPU attributes among various peripheral clock domains. The synchronization process is started when the PPU module is unlocked, by writing correct unlock key to the PPUn_UNLOCK register. On completion of unlocking sequence, the PPU lock status PPUn_ST:LST bit is set to '0'. If the PPUn_CTR:DMAEN bit is already set, the DMA request is also generated. This allows configuring the PPU registers through DMA transfers. Once PPU registers are configured, the PPU module needs to be locked, to prevent accidental change of PPU attributes. The locking of PPU module is done be writing correct lock key to the PPUn_UNLOCK register. The PPU Lock Status PPUn_ST:LST bit is set to '1'. Once the synchronization process is complete, the PPUn_ST:PSA bit becomes '1', indicating that the new PPU attribute settings are valid in all peripheral clock domains.

DMA interface

The PPU provides DMA request to initiate DMA transfers. The DMA request is generated when the PPU module is unlocked. The DMA enable bit PPUn_CTR:DMAEN needs to be set for the DMA request to be generated.

21.2 PPU registers

The PPU module contains various registers to configure its operation, to monitor its status, and to program the read/access attributes for peripherals and GPIO channels. The PPU module is allocated 1 KB of MCU's address space for mapping the Configuration and Control/Status Registers (i.e. CSRs) of PPU. The address area allocated to the PPU and the Control and Status Registers in PPU are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of PPU

The following registers are available for the PPU:

- PPU Peripheral Read Attribute Register (PPUn_PR0~15)
- PPU Peripheral Read Attribute Set Register (PPUn_PRS0~15)
- PPU Peripheral Read Attribute Clear Register (PPUn_PRC0~15)
- PPU Peripheral Access Attribute Register (PPUn_PA0~15)
- PPU Peripheral Access Attribute Set Register (PPUn_PAS0~15)
- PPU Peripheral Access Attribute Clear Register (PPUn_PAC0~15)
- PPU GPIO Access Attribute Register (PPUn_GA0~15)
- PPU GPIO Access Attribute Set Register (PPUn_GAS0~15)
- PPU GPIO Access Attribute Clear Register (PPUn_GAC0~15)
- PPU Status Register (PPUn_ST)
- PPU Control Register (PPUn_CTR)
- PPU Unlock Register (PPUn_UNLOCK)

Memory layout of PPU registers

Table 21-1. Memory layout of PPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	PPUn_PRS1 00000000 00000000 00000000 00000000				PPUn_PRS0 00000000 00000000 00000000 00000000			
0x00000008	PPUn_PRS3 00000000 00000000 00000000 00000000				PPUn_PRS2 00000000 00000000 00000000 00000000			
0x00000010	PPUn_PRS5 00000000 00000000 00000000 00000000				PPUn_PRS4 00000000 00000000 00000000 00000000			
0x00000018	PPUn_PRS7 00000000 00000000 00000000 00000000				PPUn_PRS6 00000000 00000000 00000000 00000000			
0x00000020	PPUn_PRS9 00000000 00000000 00000000 00000000				PPUn_PRS8 00000000 00000000 00000000 00000000			
0x00000028	PPUn_PRS11 00000000 00000000 00000000 00000000				PPUn_PRS10 00000000 00000000 00000000 00000000			
0x00000030	PPUn_PRS13 00000000 00000000 00000000 00000000				PPUn_PRS12 00000000 00000000 00000000 00000000			
0x00000038	PPUn_PRS15 00000000 00000000 00000000 00000000				PPUn_PRS14 00000000 00000000 00000000 00000000			
0x00000040	PPUn_PAS1 00000000 00000000 00000000 00000000				PPUn_PAS0 00000000 00000000 00000000 00000000			

Table 21-1. Memory layout of PPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000048	PPUn_PAS3 00000000 00000000 00000000 00000000				PPUn_PAS2 00000000 00000000 00000000 00000000			
0x00000050	PPUn_PAS5 00000000 00000000 00000000 00000000				PPUn_PAS4 00000000 00000000 00000000 00000000			
0x00000058	PPUn_PAS7 00000000 00000000 00000000 00000000				PPUn_PAS6 00000000 00000000 00000000 00000000			
0x00000060	PPUn_PAS9 00000000 00000000 00000000 00000000				PPUn_PAS8 00000000 00000000 00000000 00000000			
0x00000068	PPUn_PAS11 00000000 00000000 00000000 00000000				PPUn_PAS10 00000000 00000000 00000000 00000000			
0x00000070	PPUn_PAS13 00000000 00000000 00000000 00000000				PPUn_PAS12 00000000 00000000 00000000 00000000			
0x00000078	PPUn_PAS15 00000000 00000000 00000000 00000000				PPUn_PAS14 00000000 00000000 00000000 00000000			
0x00000080	PPUn_GAS1 00000000 00000000 00000000 00000000				PPUn_GAS0 00000000 00000000 00000000 00000000			
0x00000088	PPUn_GAS3 00000000 00000000 00000000 00000000				PPUn_GAS2 00000000 00000000 00000000 00000000			
0x00000090	PPUn_GAS5 00000000 00000000 00000000 00000000				PPUn_GAS4 00000000 00000000 00000000 00000000			
0x00000098	PPUn_GAS7 00000000 00000000 00000000 00000000				PPUn_GAS6 00000000 00000000 00000000 00000000			
0x000000A0	PPUn_GAS9 00000000 00000000 00000000 00000000				PPUn_GAS8 00000000 00000000 00000000 00000000			
0x000000A8	PPUn_GAS11 00000000 00000000 00000000 00000000				PPUn_GAS10 00000000 00000000 00000000 00000000			
0x000000B0	PPUn_GAS13 00000000 00000000 00000000 00000000				PPUn_GAS12 00000000 00000000 00000000 00000000			
0x000000B8	PPUn_GAS15 00000000 00000000 00000000 00000000				PPUn_GAS14 00000000 00000000 00000000 00000000			
0x000000C0	PPUn_PRC1 00000000 00000000 00000000 00000000				PPUn_PRC0 00000000 00000000 00000000 00000000			
0x000000C8	PPUn_PRC3 00000000 00000000 00000000 00000000				PPUn_PRC2 00000000 00000000 00000000 00000000			
0x000000D0	PPUn_PRC5 00000000 00000000 00000000 00000000				PPUn_PRC4 00000000 00000000 00000000 00000000			
0x000000D8	PPUn_PRC7 00000000 00000000 00000000 00000000				PPUn_PRC6 00000000 00000000 00000000 00000000			

Table 21-1. Memory layout of PPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000000E0	PPUn_PRC9 00000000 00000000 00000000 00000000				PPUn_PRC8 00000000 00000000 00000000 00000000			
0x000000E8	PPUn_PRC11 00000000 00000000 00000000 00000000				PPUn_PRC10 00000000 00000000 00000000 00000000			
0x000000F0	PPUn_PRC13 00000000 00000000 00000000 00000000				PPUn_PRC12 00000000 00000000 00000000 00000000			
0x000000F8	PPUn_PRC15 00000000 00000000 00000000 00000000				PPUn_PRC14 00000000 00000000 00000000 00000000			
0x00000100	PPUn_PAC1 00000000 00000000 00000000 00000000				PPUn_PAC0 00000000 00000000 00000000 00000000			
0x00000108	PPUn_PAC3 00000000 00000000 00000000 00000000				PPUn_PAC2 00000000 00000000 00000000 00000000			
0x00000110	PPUn_PAC5 00000000 00000000 00000000 00000000				PPUn_PAC4 00000000 00000000 00000000 00000000			
0x00000118	PPUn_PAC7 00000000 00000000 00000000 00000000				PPUn_PAC6 00000000 00000000 00000000 00000000			
0x00000120	PPUn_PAC9 00000000 00000000 00000000 00000000				PPUn_PAC8 00000000 00000000 00000000 00000000			
0x00000128	PPUn_PAC11 00000000 00000000 00000000 00000000				PPUn_PAC10 00000000 00000000 00000000 00000000			
0x00000130	PPUn_PAC13 00000000 00000000 00000000 00000000				PPUn_PAC12 00000000 00000000 00000000 00000000			
0x00000138	PPUn_PAC15 00000000 00000000 00000000 00000000				PPUn_PAC14 00000000 00000000 00000000 00000000			
0x00000140	PPUn_GAC1 00000000 00000000 00000000 00000000				PPUn_GAC0 00000000 00000000 00000000 00000000			
0x00000148	PPUn_GAC3 00000000 00000000 00000000 00000000				PPUn_GAC2 00000000 00000000 00000000 00000000			
0x00000150	PPUn_GAC5 00000000 00000000 00000000 00000000				PPUn_GAC4 00000000 00000000 00000000 00000000			
0x00000158	PPUn_GAC7 00000000 00000000 00000000 00000000				PPUn_GAC6 00000000 00000000 00000000 00000000			
0x00000160	PPUn_GAC9 00000000 00000000 00000000 00000000				PPUn_GAC8 00000000 00000000 00000000 00000000			
0x00000168	PPUn_GAC11 00000000 00000000 00000000 00000000				PPUn_GAC10 00000000 00000000 00000000 00000000			
0x00000170	PPUn_GAC13 00000000 00000000 00000000 00000000				PPUn_GAC12 00000000 00000000 00000000 00000000			

Table 21-1. Memory layout of PPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000178	PPUn_GAC15 00000000 00000000 00000000 00000000				PPUn_GAC14 00000000 00000000 00000000 00000000			
0x00000180	PPUn_PR1 00000000 00000000 00000000 00000000				PPUn_PR0 00000000 00000000 00000000 00000000			
0x00000188	PPUn_PR3 00000000 00000000 00000000 00000000				PPUn_PR2 00000000 00000000 00000000 00000000			
0x00000190	PPUn_PR5 00000000 00000000 00000000 00000000				PPUn_PR4 00000000 00000000 00000000 00000000			
0x00000198	PPUn_PR7 00000000 00000000 00000000 00000000				PPUn_PR6 00000000 00000000 00000000 00000000			
0x000001A0	PPUn_PR9 00000000 00000000 00000000 00000000				PPUn_PR8 00000000 00000000 00000000 00000000			
0x000001A8	PPUn_PR11 00000000 00000000 00000000 00000000				PPUn_PR10 00000000 00000000 00000000 00000000			
0x000001B0	PPUn_PR13 00000000 00000000 00000000 00000000				PPUn_PR12 00000000 00000000 00000000 00000000			
0x000001B8	PPUn_PR15 00000000 00000000 00000000 00000000				PPUn_PR14 00000000 00000000 00000000 00000000			
0x000001C0	PPUn_PA1 00000000 00000000 00000000 00000000				PPUn_PA0 00000000 00000000 00000000 00000000			
0x000001C8	PPUn_PA3 00000000 00000000 00000000 00000000				PPUn_PA2 00000000 00000000 00000000 00000000			
0x000001D0	PPUn_PA5 00000000 00000000 00000000 00000000				PPUn_PA4 00000000 00000000 00000000 00000000			
0x000001D8	PPUn_PA7 00000000 00000000 00000000 00000000				PPUn_PA6 00000000 00000000 00000000 00000000			
0x000001E0	PPUn_PA9 00000000 00000000 00000000 00000000				PPUn_PA8 00000000 00000000 00000000 00000000			
0x000001E8	PPUn_PA11 00000000 00000000 00000000 00000000				PPUn_PA10 00000000 00000000 00000000 00000000			
0x000001F0	PPUn_PA13 00000000 00000000 00000000 00000000				PPUn_PA12 00000000 00000000 00000000 00000000			
0x000001F8	PPUn_PA15 00000000 00000000 00000000 00000000				PPUn_PA14 00000000 00000000 00000000 00000000			
0x00000200	PPUn_GA1 00000000 00000000 00000000 00000000				PPUn_GA0 00000000 00000000 00000000 00000000			
0x00000208	PPUn_GA3 00000000 00000000 00000000 00000000				PPUn_GA2 00000000 00000000 00000000 00000000			

Table 21-1. Memory layout of PPU registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000210	PPUn_GA5 00000000 00000000 00000000 00000000				PPUn_GA4 00000000 00000000 00000000 00000000			
0x00000218	PPUn_GA7 00000000 00000000 00000000 00000000				PPUn_GA6 00000000 00000000 00000000 00000000			
0x00000220	PPUn_GA9 00000000 00000000 00000000 00000000				PPUn_GA8 00000000 00000000 00000000 00000000			
0x00000228	PPUn_GA11 00000000 00000000 00000000 00000000				PPUn_GA10 00000000 00000000 00000000 00000000			
0x00000230	PPUn_GA13 00000000 00000000 00000000 00000000				PPUn_GA12 00000000 00000000 00000000 00000000			
0x00000238	PPUn_GA15 00000000 00000000 00000000 00000000				PPUn_GA14 00000000 00000000 00000000 00000000			
0x00000240	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				PPUn_UNLOCK 00000000 00000000 00000000 00000000			
0x00000248	PPUn_CTR XXXXXXXX 00000000 00000000 00000000				PPUn_ST XXXXXXXX XXXXXXXX 00000001 00000001			

21.2.1 PPU Peripheral Read Attribute Register (PPUn_PR0~15)

The PPU Peripheral Read Attribute Registers provide read attribute for peripherals. The software can use these registers to directly set/clear the individual peripherals read attributes. Apart from this register, the PPUn_PR0~15 bits can be set and cleared using PPUn_PRS0~15 and PPUn_PRC0~15 registers that are explained later. As one register accommodates read attributes of 32 peripherals, PPU provides 16 such registers (PPUn_PR0 to PPUn_PR15) to support up to 512 peripherals. The PPUn_PR0 provides read attributes for peripherals 0 to 31. PPUn_PR1 provides read attribute for peripherals 32 to 63. The last register PPUn_PR15 provides read attribute for peripherals 480 to 511. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_PR0 is explained here.

PPU Peripheral Read Attribute Register (PPUn_PR0)

Figure 21-2. PPU Peripheral Read Attribute Register (PPUn_PR0)

PPUn_PR0																															
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**
PR[31]	PR[30]	PR[29]	PR[28]	PR[27]	PR[26]	PR[25]	PR[24]	PR[23]	PR[22]	PR[21]	PR[20]	PR[19]	PR[18]	PR[17]	PR[16]	PR[15]	PR[14]	PR[13]	PR[12]	PR[11]	PR[10]	PR[9]	PR[8]	PR[7]	PR[6]	PR[5]	PR[4]	PR[3]	PR[2]	PR[1]	PR[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-2. PPU Peripheral Read Attribute Register (PPUn_PR0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PR	Peripheral Read Attribute The individual bits of this register provide set/clear of the read attribute of peripheral resource channels. '0': Clears the corresponding read attribute of peripheral resource channel '1': Sets the corresponding read attribute of peripheral resource channel Reading provides set/clear status of read attribute of peripheral resource channels. The bits in this register correspond to peripheral resources [31:0].

21.2.2 PPU Peripheral Read Attribute Set Register (PPUn_PRS0~15)

The software can set the individual peripheral read attribute using PPUn_PRS0~15 registers. Writing '1' to the bits of this register sets the corresponding bit in the Peripheral Read Attribute Register (PPUn_PR0~15). As PPU supports 512 peripherals it provides 16 Peripheral Read Attribute Set Registers. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_PRS0 is explained below.

PPU Peripheral Read Attribute Set Register (PPUn_PRS0)

Figure 21-3. PPU Peripheral Read Attribute Set Register (PPUn_PRS0)

PPUn_PRS0																															
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**
PRS[31]	PRS[30]	PRS[29]	PRS[28]	PRS[27]	PRS[26]	PRS[25]	PRS[24]	PRS[23]	PRS[22]	PRS[21]	PRS[20]	PRS[19]	PRS[18]	PRS[17]	PRS[16]	PRS[15]	PRS[14]	PRS[13]	PRS[12]	PRS[11]	PRS[10]	PRS[9]	PRS[8]	PRS[7]	PRS[6]	PRS[5]	PRS[4]	PRS[3]	PRS[2]	PRS[1]	PRS[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-3. PPU Peripheral Read Attribute Set Register (PPUn_PRS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PRS	Software Set Peripheral Read Attribute '0': No effect '1': Sets the corresponding read attribute of peripheral resource channel This register always reads '0'. The bits in this register correspond to peripheral resources [31:0].

21.2.3 PPU Peripheral Read Attribute Clear Register (PPUn_PRC0~15)

The software can clear the individual peripheral read attribute using PPUn_PRC0~15 registers. Writing '1' to the bits of this register clears the corresponding bit in the Peripheral Read Attribute Register (PPUn_PRC0~15). As PPU supports 512 peripherals, it provides 16 Peripheral Read Attribute Clear Registers. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_PRC0 is explained below.

PPU Peripheral Read Attribute Clear Register (PPUn_PRC0)

Figure 21-4. PPU Peripheral Read Attribute Clear Register (PPUn_PRC0)

PPUn_PRC0																															
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**
PRC[31]	PRC[30]	PRC[29]	PRC[28]	PRC[27]	PRC[26]	PRC[25]	PRC[24]	PRC[23]	PRC[22]	PRC[21]	PRC[20]	PRC[19]	PRC[18]	PRC[17]	PRC[16]	PRC[15]	PRC[14]	PRC[13]	PRC[12]	PRC[11]	PRC[10]	PRC[9]	PRC[8]	PRC[7]	PRC[6]	PRC[5]	PRC[4]	PRC[3]	PRC[2]	PRC[1]	PRC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-4. PPU Peripheral Read Attribute Clear Register (PPUn_PRC0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PRC	Software Clear Peripheral Read Attribute '0': No effect '1': Clears the corresponding read attribute of peripheral resource channel This register always reads '0'. The bits in this register correspond to peripheral resources [31:0].

21.2.4 PPU Peripheral Access Attribute Register (PPUn_PA0~15)

The PPU Peripheral Access Attribute Registers provide access attribute for peripherals. The software can use these registers to directly set/clear the individual peripherals access attributes. Apart from this register, the PPUn_PA0~15 bits can be set, and cleared by using PPUn_PAS0~15 and PPUn_PAC0~15 registers that are explained later. As one register accommodates access attributes of 32 peripherals, PPU provides 16 such registers (PPUn_PA0 to PPUn_PA15) to support upto 512 peripherals. The PPUn_PA0 provides access attributes for peripherals 0 to 31. PPUn_PA1 provides access attribute for peripherals 32 to 63. The last register PPUn_PA15 provides access attribute for peripherals 480 to 511. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_PA0 is explained here.

PPU Peripheral Access Attribute Register (PPUn_PA0)

Figure 21-5. PPU Peripheral Access Attribute Register (PPUn_PA0)

PPUn_PA0																															
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**
PA[31]	PA[30]	PA[29]	PA[28]	PA[27]	PA[26]	PA[25]	PA[24]	PA[23]	PA[22]	PA[21]	PA[20]	PA[19]	PA[18]	PA[17]	PA[16]	PA[15]	PA[14]	PA[13]	PA[12]	PA[11]	PA[10]	PA[9]	PA[8]	PA[7]	PA[6]	PA[5]	PA[4]	PA[3]	PA[2]	PA[1]	PA[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-5. PPU Peripheral Access Attribute Register (PPUn_PA0)

Bit Position	Bit Field Name	Bit Description
[31:0]	PA	Peripheral Access Attribute The individual bits of this register provide set/clear of the access attribute of peripheral resource channels. '0': Clears the corresponding access attribute of peripheral resource channels '1': Sets the corresponding access attribute of peripheral resource channels Reading provides set/clear status of access attribute of peripheral resource channels. The bits in this register correspond to peripheral resources [31:0].

21.2.5 PPU Peripheral Access Attribute Set Register (PPUn_PAS0~15)

The software can set the individual peripheral access attribute using PPUn_PAS0~15 registers. Writing '1' to the bits of this register sets the corresponding bit in the Peripheral Access Attribute Register (PPUn_PA0~15). As PPU supports 512 peripherals it provides 16 Peripheral Access Attribute Set Registers. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_PAS0 is explained below.

PPU Peripheral Access Attribute Set Register (PPUn_PAS0)

Figure 21-6. PPU Peripheral Access Attribute Set Register (PPUn_PAS0)

PPUn_PAS0																															
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**
PAS[31]	PAS[30]	PAS[29]	PAS[28]	PAS[27]	PAS[26]	PAS[25]	PAS[24]	PAS[23]	PAS[22]	PAS[21]	PAS[20]	PAS[19]	PAS[18]	PAS[17]	PAS[16]	PAS[15]	PAS[14]	PAS[13]	PAS[12]	PAS[11]	PAS[10]	PAS[9]	PAS[8]	PAS[7]	PAS[6]	PAS[5]	PAS[4]	PAS[3]	PAS[2]	PAS[1]	PAS[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-6. PPU Peripheral Access Attribute Set Register (PPUn_PAS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PAS	Software Set Peripheral Access Attribute '0': No effect '1': Sets the corresponding access attribute of peripheral resource channels This register always reads '0'. The bits in this register correspond to peripheral resources [31:0].

21.2.6 PPU Peripheral Access Attribute Clear Register (PPUn_PAC0~15)

The software can clear the individual peripheral access attribute by using PPUn_PAC0~15 registers. Writing '1' to the bits of this register clears the corresponding bit in the Peripheral Access Attribute Register (PPUn_PA0~15). As PPU supports 512 peripherals it provides 16 Peripheral Access Attribute Clear Registers. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_PAC0 is explained below.

PPU Peripheral Access Attribute Clear Register (PPUn_PAC0)

Figure 21-7. PPU Peripheral Access Attribute Clear Register (PPUn_PAC0)

PPUn_PAC0																															
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**
PAC[31]	PAC[30]	PAC[29]	PAC[28]	PAC[27]	PAC[26]	PAC[25]	PAC[24]	PAC[23]	PAC[22]	PAC[21]	PAC[20]	PAC[19]	PAC[18]	PAC[17]	PAC[16]	PAC[15]	PAC[14]	PAC[13]	PAC[12]	PAC[11]	PAC[10]	PAC[9]	PAC[8]	PAC[7]	PAC[6]	PAC[5]	PAC[4]	PAC[3]	PAC[2]	PAC[1]	PAC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-7. PPU Peripheral Access Attribute Clear Register (PPUn_PAC0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	PAC	Software Clear Peripheral Access Attribute '0': No effect '1': Clears the corresponding access attribute of peripheral resource channels This register always reads '0'. The bits in this register correspond to peripheral resources [31:0].

21.2.7 PPU GPIO Access Attribute Register (PPUn_GA0~15)

The PPU GPIO Access Attribute Registers provide access attribute for GPIO channels. The software can use these registers to directly set/clear the individual GPIO channels access attributes. Apart from this register, the PPUn_GA0~15 bits can be set, and cleared by using PPUn_GAS0~15n and PPUn_GAC0~15 registers that are explained later. As one register accommodates access attributes of 32 GPIO channels, PPU provides 16 such registers (PPUn_GA0 to PPUn_GA15) to support upto 512 GPIO channels. The PPUn_GA0 provides access attributes for GPIO channels 31 to 0. PPUn_GA1 provides access attribute for GPIO channels 63 to 32. The last register PPUn_GA15 provides access attribute for GPIO channels 511 to 480. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_GA0 is explained here.

PPU GPIO Access Attribute Register (PPUn_GA0)

Figure 21-8. PPU GPIO Access Attribute Register (PPUn_GA0)

PPUn_GA0																															
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**
GA[31]	GA[30]	GA[29]	GA[28]	GA[27]	GA[26]	GA[25]	GA[24]	GA[23]	GA[22]	GA[21]	GA[20]	GA[19]	GA[18]	GA[17]	GA[16]	GA[15]	GA[14]	GA[13]	GA[12]	GA[11]	GA[10]	GA[9]	GA[8]	GA[7]	GA[6]	GA[5]	GA[4]	GA[3]	GA[2]	GA[1]	GA[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-8. PPU GPIO Access Attribute Register (PPUn_GA0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	GA	GPIO Access Attribute The individual bits of this register provide set/clear of the access attribute of GPIO channels. '0': Clears the corresponding access attribute of GPIO channels '1': Sets the corresponding access attribute of GPIO channels Reading provides set/clear status of access attribute of GPIO channels. The bits in this register correspond to GPIO channels [31:0].

21.2.8 PPU GPIO Access Attribute Set Register (PPUn_GAS0~15)

The software can set the individual GPIO channel access attribute by using PPUn_GAS0~15 registers. Writing '1' to the bits of this register sets the corresponding bit in the GPIO Channel Access Attribute Register (PPUn_GA0~15). As PPU supports 512 GPIO channels it provides 16 GPIO Access Attribute Set Registers. Register PPUn_GAS0 is explained below. This register can be written only in privileged mode and when PPUn_ST:LST = '0'.

PPU GPIO Access Attribute Set Register (PPUn_GAS0)

Figure 21-9. PPU GPIO Access Attribute Set Register (PPUn_GAS0)

PPUn_GAS0																															
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**
GAS[31]	GAS[30]	GAS[29]	GAS[28]	GAS[27]	GAS[26]	GAS[25]	GAS[24]	GAS[23]	GAS[22]	GAS[21]	GAS[20]	GAS[19]	GAS[18]	GAS[17]	GAS[16]	GAS[15]	GAS[14]	GAS[13]	GAS[12]	GAS[11]	GAS[10]	GAS[9]	GAS[8]	GAS[7]	GAS[6]	GAS[5]	GAS[4]	GAS[3]	GAS[2]	GAS[1]	GAS[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-9. PPU GPIO Access Attribute Set Register (PPUn_GAS0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	GAS	Software Set GPIO Access Attribute '0': No effect '1': Sets the corresponding access attribute of GPIO channels This register always reads '0'. The bits in this register correspond to GPIO channels [31:0].

21.2.9 PPU GPIO Access Attribute Clear Register (PPUn_GAC0~15)

The software can clear the individual GPIO channel access attribute by using PPUn_GAC0~15 registers. Writing '1' to the bits of this register clears the corresponding bit in the GPIO Channel Access Attribute Register (PPUn_GA0~15). As PPU supports 512 GPIO channels, it provides 16 GPIO Access Attribute Clear Registers. This register can be written only in privileged mode and when PPUn_ST:LST = '0'. Register PPUn_GAC0 is explained below.

PPU GPIO Access Attribute Clear Register (PPUn_GAC0)

Figure 21-10. PPU GPIO Access Attribute Clear Register (PPUn_GAC0)

PPUn_GAC0																															
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**
GAC[31]	GAC[30]	GAC[29]	GAC[28]	GAC[27]	GAC[26]	GAC[25]	GAC[24]	GAC[23]	GAC[22]	GAC[21]	GAC[20]	GAC[19]	GAC[18]	GAC[17]	GAC[16]	GAC[15]	GAC[14]	GAC[13]	GAC[12]	GAC[11]	GAC[10]	GAC[9]	GAC[8]	GAC[7]	GAC[6]	GAC[5]	GAC[4]	GAC[3]	GAC[2]	GAC[1]	GAC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-10. PPU GPIO Access Attribute Clear Register (PPUn_GAC0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	GAC	Software Clear GPIO Access Attribute '0': No effect '1': Clears the corresponding access attribute of GPIO channels This register always reads '0'. The bits in this register correspond to GPIO channels [31:0].

21.2.10 PPU Status Register (PPUn_ST)

The software can use this register to read PPU module locked or unlocked status, and status of PPU attributes in peripheral resources and GPIO channels.

PPU Status Register (PPUn_ST)

Figure 21-11. PPU Status Register (PPUn_ST)

PPUn_ST																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	PSA	read0	read0	read0	read0	read0	read0	read0	LST
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0	R
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Table 21-11. PPU Status Register (PPUn_ST) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	reserved	-
[15:9]	read0	-
[8]	PSA	PPU Settings Applied Status of PPU attributes in peripheral resources and GPIO channels. '0': PPU attributes are still being applied in peripheral resources and GPIO channels '1': PPU attributes are stable in peripheral resources and GPIO channels PPU attributes cannot be configured before PPUn_ST:LST = '0'.
[7:1]	read0	-
[0]	LST	PPU Lock Status Lock status for PPU. '0': PPU module is unlocked '1': PPU module is locked The PPU attributes should be configured once PPU module is unlocked.

21.2.11 PPU Control Register (PPUn_CTR)

The PPU Control Register contains information about DMA enable, control for access to test register and enable function to non-existent address violation error. This register can be written only in privileged mode and when PPUn_ST:LST = '0'.

PPU Control Register (PPUn_CTR)

Figure 21-12. PPU Control Register (PPUn_CTR)

PPUn_CTR																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	NEAV	read0	read0	read0	read0	read0	read0	read0	PTST	read0	read0	read0	read0	read0	read0	read0	DMAEN
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP	R0	R0	R0	R0	R0	R0	R0	RWP	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-12. PPU Control Register (PPUn_CTR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	reserved	-
[23:17]	read0	-
[16]	NEAV	Non-Existent Address Violation '0': No error to BECU is flagged in case of access to a non-existent address space '1': Error to BECU is flagged in case of access to a non-existent address space The error to BECU is always generated for non-existent addresses outside the defined register area for PPU module. Errors which are generated for addresses in a certain range can be enabled/disabled with this bit. This address range includes the read/access bits along with the set/clear bits.
[15:9]	read0	-
[8]	PTST	Protection of Test Registers '0': Protects all test registers on the device, access in all modes is forbidden '1': No special protection of test registers on the device done,
[7:1]	read0	-

Table 21-12. PPU Control Register (PPUn_CTR) bits

Bit Position	Bit Field Name	Bit Description
[0]	DMAEN	DMA Enable DMA access enable. '0': DMA channel is disabled '1': DMA channel is enabled,

21.2.12 PPU Unlock Register (PPUn_UNLOCK)

The software must use this register to lock/unlock the PPU registers for write access. This register can be written only in privileged mode.

PPU Unlock Register (PPUn_UNLOCK)

Figure 21-13. PPU Unlock Register (PPUn_UNLOCK)

PPUn_UNLOCK																													
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-13. PPU Unlock Register (PPUn_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	PPU Unlock The PPU Unlock Register protects the PPU module from being modified accidently by software. The PPU registers cannot be written until this register has been written with a specific correct value. The correct value for unlocking (0xACC5BB01) can be written only in privilege mode. The read to this register always returns '0'. To lock the PPU module (0xBB0B10C1), software must write another value specific to lock. Writing to any PPU module regsiters except Unlock Register, while the module is locked (i.e., PPUn_ST:LST bit is '1') is illegal and causes protection error. The software should unlock the PPU module in order to modify the attribute settings. The software should lock the PPU module for the attribute settings to take effect.

Note: The software must always make a 64-bit or 32-bit access to this register. An 8-bit or a 16-bit access to this register causes protection error.

21.3 Operation of PPU

This chapter describes the operation of PPU.

PPU protection attributes

This device facilitates three types of protection attributes for protection against illegal access by software, to the registers of peripherals and GPIO module. These attributes include read, access, and enable. PPU provides read and access attributes for peripherals and access attributes for GPIO module. The PPU attributes may not be implemented in each of the peripherals. The enable attribute is provided by respective Bus Support Unit (BSU) on which the peripherals are connected. The Table 21-14 explains each of the attributes when available as an individual or as a group with other attributes in the peripherals and GPIO module.

Table 21-14. PPU protection attributes

Enable	Access	Read	PPU protection level
-	-	'1'	The read attribute allows software to read peripheral registers in non-privilged mode. It also allows access of peripheral registers in privileged mode.
-	'1'	-	The access attribute allows software to read and write peripheral and GPIO module registers in both non-privileged and privileged mode.
'1'	-	-	The peripheral and GPIO module are enabled or are available. Based on the presence of PPU read and/or PPU access attribute allows software to read and/or write peripheral and GPIO module registers in both non-privileged and privileged mode.
'0'	x	x	The peripheral and GPIO modules are disabled or are not available.
'1'	'0'	'0'	Allows software to read and write peripheral registers in privileged mode. The software is not permitted to read and write peripheral registers in non-privileged mode.
'1'	'0'	-	Allows software to read GPIO module registers in non-privileged mode. Read and write access in privileged mode are permitted.
'1'	'0'	'1'	Allows software to read peripheral registers in non-privileged mode. Registers for which the read access triggers some action, or has the consequence of writing cannot be read in non-privileged mode. The software is not permitted to write peripheral registers in non-privileged mode. Read and write accesses in privileged mode are permitted.
'1'	'1'	x	Allows software to read and write in privileged and non-privileged modes.
-	-	-	If read, access, and enable attributes are not available, allows software to write peripheral and GPIO module registers only in privileged mode. Read of registers is permitted in both non-privileged and privileged modes. System modules in particular, use such a protection.

Legends:

'-' indicates that the attribute is not available in respective resource

'x' indicates that the attribute is don’t care, due to precedence by other attribute

'1' or '0' indicates set and clear of attribute in respective resource

Note 1:

When an access attribute for a particular channel is set, the value of read attribute of that channel can be ignored. Depending on the usage, the software needs to set the attributes of the respective peripherals and GPIO channels. The PPU module supports 512 channels for peripherals. Each channel for peripheral provides read and access attribute. The PPU module also supports 512 channels for GPIO module. Each channel for GPIO module provides access attribute.

Note 2:

For more details on the enable attribute, refer to 37. Bus Support Unit.

Note 3:

For more details on channel allocation to peripherals and GPIO module available in FCR4 Cluster, refer to device specific datasheet.

21.4 Notes on using PPU

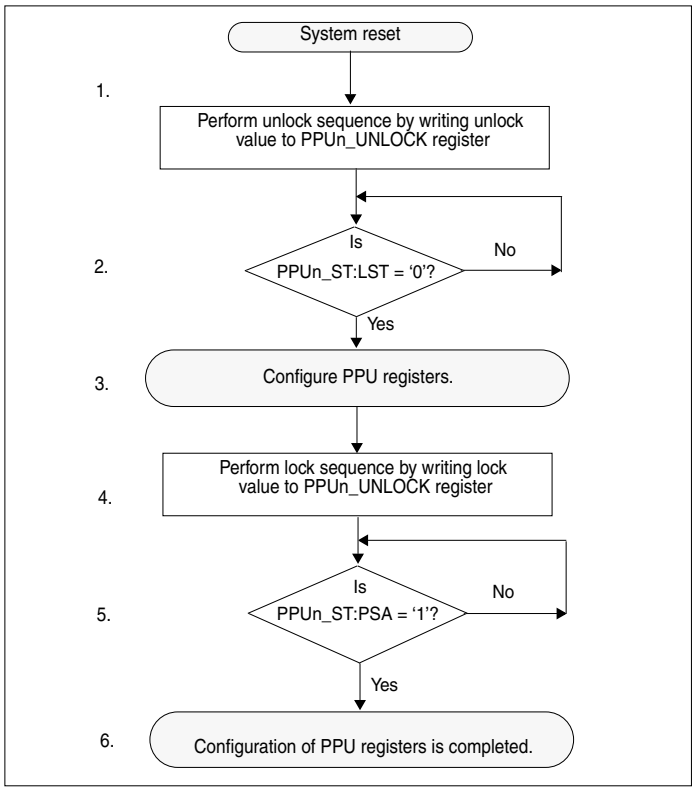
This section is the 'programmer's guide', which lists the usage notes for programming the PPU module. Programmers are supposed to read these guidelines before programming the PPU module.

General usage notes

Reserved bits return undefined values. The software programs shall be independent of the values read from the reserved register bits.

Steps in programming PPU attributes for a resource

Figure 21-14. Programmer's flowchart



The general steps a programmer should follow while using the PPU module are as follows:

1. After the system reset, the software performs the unlocking sequence by writing unlock value to the PPUn_UNLOCK register. The unlock and lock values are provided in the device specific datasheet.
2. The software polls the lock status bit, PPUn_ST:LST, on completion of unlock sequence this bit is made '0'.
3. The software configures the PPU registers. The unlocking process, also triggers DMA transfer request, provided DMA channel enabled is set (PPUn_CTR:DMAEN = '1'). This allows configuring PPU registers through DMA access.
4. Once the registers of PPU are configured, the software performs locking sequence by writing lock value to the PPUn_UNLOCK register.
5. The PPU module is locked and the software now polls for PPUn_ST:PSA bit, when this bit is '1'. This indicates that the PPU settings are already applied to the respective peripheral resources and GPIO channels.
6. Configuration of PPU registers is completed, and new PPU attributes are applicable to all corresponding PPU protected peripherals.

22. Bus Error Collection Unit

This chapter explains the function and operation of the Bus Error Collection Unit (BECU).

22.1 Outline of Bus Error Collection Unit (BECU)

The BECU module monitors the RBUS/eRBUS for slave responses. If any slave on the bus responds with an error, it collects all information about the bus access which caused the error and stores the collected information in its internal registers. The features of BECU module are described in this section.

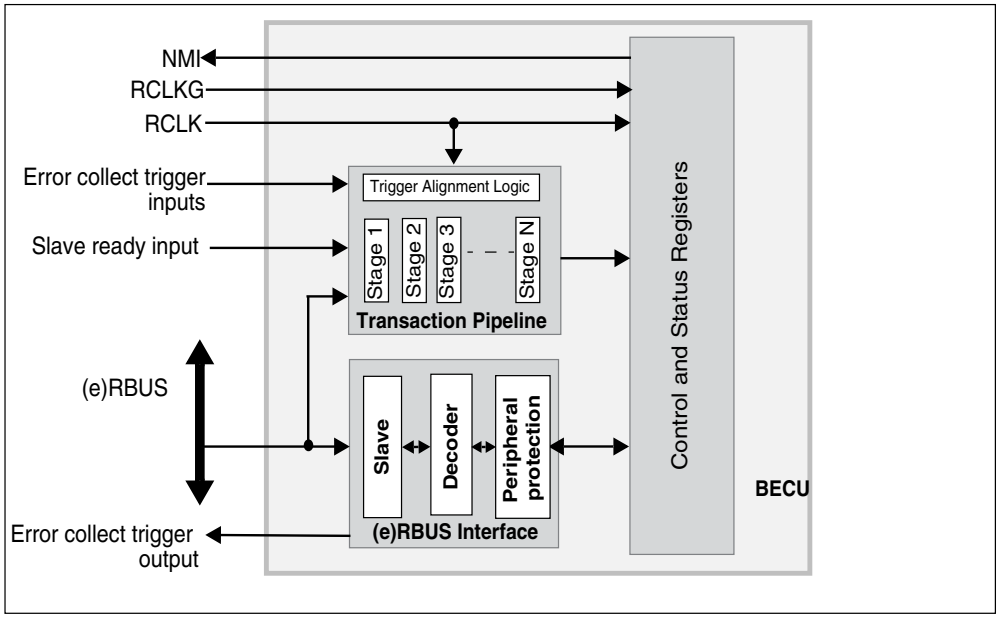
Features of Bus Error Collection Unit (BECU)

The BECU module monitors the (e)RBUS for bus errors. When a bus error occurs, it registers the information about the error in its registers and also triggers a Non-Maskable Interrupt (NMI) to the CPU. The CPU may later use this information for error diagnostic purposes. The features of the BECU are:

- It supports 16-bit RBUS as well as its extended variant, the 64-bit eRBUS
- It collects information about the bus access that caused the error
- On occurrence of an error event, the BECU can optionally propagate an NMI to the CPU
- Information collected about the error is accessible to the CPU through BECU's register set

Block diagram of BECU

Figure 22-1. Block diagram of BECU



22.2 BECU registers

The BECU module contains various Configuration and Status Registers (CSRs) as well as registers to read the information it has collected from the (e)RBUS when an error has occured. The BECU module is allocated 1 KB of the MCU's address space for mapping the CSRs of the BECU. The CSRs are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of BECU

The following registers are available for each instance of BECU:

- BECU Control Register - L (BECUn_CTRL)
- BECU Control Register - H (BECUn_CTRH)
- BECU Address Register - L (BECUn_ADDRL)
- BECU Address Register - H (BECUn_ADDRH)
- BECU Data Register - LL (BECUn_DATALL)
- BECU Data Register - LH (BECUn_DATALH)
- BECU Data Register - HL (BECUn_DATAHL)
- BECU Data Register - HH (BECUn_DATAHH)
- BECU Master ID Register (BECUn_MASTERID)
- BECU Module ID Register - L (BECUn_MIDL)
- BECU Module ID Register - H (BECUn_MIDH)
- BECU NMI Enable Register - (BECUn_NMIEN)

Memory layout of BECU registers

Table 22-1. Memory layout of BECU registers

Offset	+1	+0
0x00000000	BECUn_CTRL 00000000 00000000	
0x00000002	BECUn_CTRH 00000000 00000000	
0x00000004	BECUn_ADDRL 00000000 00000000	
0x00000006	BECUn_ADDRH 00000000 00000000	
0x00000008	BECUn_DATALL 00000000 00000000	
0x0000000A	BECUn_DATALH 00000000 00000000	
0x0000000C	BECUn_DATAHL 00000000 00000000	
0x0000000E	BECUn_DATAHH 00000000 00000000	
0x00000010	BECUn_MASTERID 00000000 00000000	

Table 22-1. Memory layout of BECU registers

Offset	+1	+0
0x00000012	BECUn_MIDL XXXXXXXX XXXXXXXX	
0x00000014	BECUn_MIDH XXXXXXXX XXXXXXXX	
0x00000016	read0 00000000 00000000	
0x00000018	BECUn_NMIEN XXXXXXXX 00000001	
0x0000001A	read0 00000000 00000000	
0x0000001C	read0 00000000 00000000	
0x0000001E	read0 00000000 00000000	

22.2.1 BECU Control Register - L (BECUn_CTRL)

The BECU Control Register - L can be used by the software to read the NMI status flag to clear the NMI status flag and/or to disable the propagation of the NMI to the Interrupt Controller. This register also contains information about the state of the privileged mode captured from the bus for the access which caused the error response.

BECU Control Register - L (BECUn_CTRL)

Figure 22-2. BECU Control Register - L (BECUn_CTRL)

BECUn_CTRL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	PROT	read0	read0	read0	read0	read0	read0	NMICL	NMI
R0	R0	R0	R0	R0	R0	R0	R	R0	R0	R0	R0	R0	R0	R0WP1	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-2. BECU Control Register - L (BECUn_CTRL) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	PROT	Protection Information This bit gives the protection information of the (e)RBUS access which caused the error response. '0': User mode access '1': Privilege mode access
[7:2]	read0	-
[1]	NMICL	Non-Maskable Interrupt Clear '0': No effect '1': Software clears the BECUn_CTRL:NMI status flag
[0]	NMI	Non-Maskable Interrupt Flag This bit indicates the status of the Non-Maskable Interrupt (NMI) generated by the BECU. '0': NMI is not pending '1': NMI is pending

22.2.2 BECU Control Register - H (BECUn_CTRH)

The BECU Control Register - H contains information about the state of the read and write strobe signals captured from the (e)RBUS for the access which caused the error response. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set.

BECU Control Register - H (BECUn_CTRH)

Figure 22-3. BECU Control Register - H (BECUn_CTRH)

BECUn_CTRH															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
WR[7]	WR[6]	WR[5]	WR[4]	WR[3]	WR[2]	WR[1]	WR[0]	RD[7]	RD[6]	RD[5]	RD[4]	RD[3]	RD[2]	RD[1]	RD[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-3. BECU Control Register - H (BECUn_CTRH) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	WR	Write Strobes This field stores the write strobes of the (e)RBUS access which caused the error response. In the case of RBUS, only the WR[1:0] bits are valid; others are treated as read0. Each bit in the 8-bit write strobe corresponds to a byte in the write data bus. When a bit is set, it implies a write access for the corresponding byte of data.
[7:0]	RD	Read Strobes This field stores the read strobes of the (e)RBUS access which caused the error response. In the case of RBUS, only the RD[1:0] bits are valid, others are treated as read0. Each bit in the 8-bit read strobe corresponds to a byte in the read data bus. When a bit is set, it implies a read access for the corresponding byte of data.

22.2.3 BECU Address Register - L (BECUn_ADDRL)

The BECU Address Register - L contains address bits [15:0] of the global address captured from the (e)RBUS from the access which caused the error response. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set.

BECU Address Register - L (BECUn_ADDRL)

Figure 22-4. BECU Address Register - L (BECUn_ADDRL)

BECUn_ADDRL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ADDR[15]	ADDR[14]	ADDR[13]	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-4. BECU Address Register - L (BECUn_ADDRL) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	ADDR	Address [15:0] This field stores the bits [15:0] of the address bus of the bus access which caused the error response. The address is relative to the bus.

22.2.4 BECU Address Register - H (BECUn_ADDRH)

The BECU Address Register - H contains the global address bits [31:16] captured from the (e)RBUS which in turn were taken from the access which caused the error response. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set.

BECU Address Register - H (BECUn_ADDRH)

Figure 22-5. BECU Address Register - H (BECUn_ADDRH)

BECUn_ADDRH															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ADDR[31]	ADDR[30]	ADDR[29]	ADDR[28]	ADDR[27]	ADDR[26]	ADDR[25]	ADDR[24]	ADDR[23]	ADDR[22]	ADDR[21]	ADDR[20]	ADDR[19]	ADDR[18]	ADDR[17]	ADDR[16]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-5. BECU Address Register - H (BECUn_ADDRH) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	ADDR	Address [31:16] This field stores the address bus bits [31:16] of the bus access which caused the error response. The address is relative to the bus.

22.2.5 BECU Data Register - LL (BECUn_DATALL)

The BECU Data Register - LL contains data bits [15:0] from the access which caused the error. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set. The software shall read the BECUn_CTRH register to discover whether the transaction was a read transaction or a write transaction. For read transactions, BECUn_DATALL contains bits [15:0] of the read data bus. For write transactions, BECUn_DATALL contains bits [15:0] of the write data bus.

BECU Data Register - LL (BECUn_DATALL)

Figure 22-6. BECU Data Register - LL (BECUn_DATALL)

BECUn_DATALL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DATA[15]	DATA[14]	DATA[13]	DATA[12]	DATA[11]	DATA[10]	DATA[9]	DATA[8]	DATA[7]	DATA[6]	DATA[5]	DATA[4]	DATA[3]	DATA[2]	DATA[1]	DATA[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-6. BECU Data Register - LL (BECUn_DATALL) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	DATA	Data [15:0] This field stores the read/write data bus bits [15:0] of the access which caused the error.

22.2.6 BECU Data Register - LH (BECUn_DATALH)

The BECU Data Register - LH contains data bits [31:16] from the access which caused the error. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set. The software shall first read the BECUn_CTRH register to discover if the transaction was a read or a write transaction. For read transactions, BECUn_DATALH contains bits [31:16] of the read data bus. For write transactions, BECUn_DATALH contains bits [31:16] of the write data bus.

BECU Data Register - LH (BECUn_DATALH)

Figure 22-7. BECU Data Register - LH (BECUn_DATALH)

BECUn_DATALH															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DATA[31]	DATA[30]	DATA[29]	DATA[28]	DATA[27]	DATA[26]	DATA[25]	DATA[24]	DATA23]	DATA[22]	DATA21]	DATA[20]	DATA[19]	DATA[18]	DATA[17]	DATA[16]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-7. BECU Data Register - LH (BECUn_DATALH)

Bit Position	Bit Field Name	Bit Description
[15:0	DATA	Data [31:16] This field stores the read/write data bus bits [31:16] of the access which caused the error response. When the BECU is connected to the RBUS, this register is read0.

22.2.7 BECU Data Register - HL (BECUn_DATAHL)

The BECU Data Register - HL contains data bits [47:32] from the access which caused the error. This register is read 0 when the BECU is connected to an RBUS. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set. The software shall first read the BECUn_CTRH register to discover if the transaction was a read or a write transaction. For read transactions, BECUn_DATAHL contains bits [47:32] of the read data bus. For write transactions, BECUn_DATAHL contains bits [47:32] of the write data bus.

BECU Data Register - HL (BECUn_DATAHL)

Figure 22-8. BECU Data Register - HL (BECUn_DATAHL)

BECUn_DATAHL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DATA[47]	DATA[46]	DATA[45]	DATA[44]	DATA[43]	DATA[42]	DATA[41]	DATA[40]	DATA[39]	DATA[38]	DATA[37]	DATA[36]	DATA[35]	DATA[34]	DATA[33]	DATA[32]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-8. BECU Data Register - HL (BECUn_DATAHL) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	DATA	Data [47:32] This field stores the read/write data bus bits [47:32] of the access which caused the error response. When the BECU is connected to the RBUS, this register is read0.

22.2.8 BECU Data Register - HH (BECUn_DATAHH)

The BECU Data Register - HH contains data bits [63:48] from the access which caused the error. This register is read 0 when the BECU is connected to an RBUS. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set. The software shall first read the BECUn_CTRH register to discover if the transaction was a read or a write transaction. For read transactions, BECUn_DATAHH contains bits [63:48] of the read data bus. For write transactions, BECUn_DATAHH contains bits [63:48] of the write data bus.

BECU Data Register - HH (BECUn_DATAHH)

Figure 22-9. BECU Data Register - HH (BECUn_DATAHH)

BECUn_DATAHH															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DATA[63]	DATA[62]	DATA[61]	DATA[60]	DATA[59]	DATA[58]	DATA[57]	DATA[56]	DATA[55]	DATA[54]	DATA[53]	DATA[52]	DATA[51]	DATA[50]	DATA[49]	DATA[48]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-9. BECU Data Register - HH (BECUn_DATAHH) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	DATA	Data [63:48] This field stores the read/write data bus bits [63:48] of the access which caused the error response. When the BECU is connected to the RBUS, this register is read0.

22.2.9 BECU Master ID Register (BECUn_MASTERID)

The BECU Master ID Register contains the MASTER-ID of the master that caused the erroneous access. This register shall be sampled by the software when the BECUn_CTRL:NMI flag is set.

BECU Master ID Register (BECUn_MASTERID)

Figure 22-10. BECU Master ID Register (BECUn_MASTERID)

BECUn_MASTERID															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MASTER[15]	MASTER[14]	MASTER[13]	MASTER[12]	MASTER[11]	MASTER[10]	MASTER[9]	MASTER[8]	MASTER[7]	MASTER[6]	MASTER[5]	MASTER[4]	MASTER[3]	MASTER[2]	MASTER[1]	MASTER[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-10. BECU Master ID Register (BECUn_MASTERID) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	MASTER	Master [15:0] This register contains the Master-ID of the (e)RBUS master which initiated the access. Unused bits are reserved and return '0' by default.

Note: For more details on list of masters, refer to the device-specific datasheet.

22.2.10 BECU Module ID Register - L (BECUn_MIDL)

The BECU Module ID Register - L provides bits [15:0] of the unique module identification number of the version of the BECU module used in the MCU.

BECU Module ID Register - L (BECUn_MIDL)

Figure 22-11. BECU Module ID Register - L (BECUn_MIDL)

BECUn_MIDL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MID[15]	MID[14]	MID[13]	MID[12]	MID[11]	MID[10]	MID[9]	MID[8]	MID[7]	MID[6]	MID[5]	MID[4]	MID[3]	MID[2]	MID[1]	MID[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 22-11. BECU Module ID Register - L (BECUn_MIDL) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	MID	Module ID [15:0] This read-only register returns bits [15:0] of the unique module identification (ID) number of the BECU module. This unique ID number identifies the version of the BECU module used in the MCU. Refer to the device-specific datasheet for the BESU module identification number.

22.2.11 BECU Module ID Register - H (BECUn_MIDH)

BECU Module ID Register - H provides bits [31:16] of the unique module identification number of the version of the BECU module used in the MCU.

BECU Module ID Register - H (BECUn_MIDH)

Figure 22-12. BECU Module ID Register - H (BECUn_MIDH)

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

Table 22-12. BECU Module ID Register - H (BECUn_MIDH) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	MID	Module ID [31:16] This read-only register returns bits [31:16] of the unique module identification (ID) number of the BECU module. This unique number identifies the version of the BECU module used in the MCU. Refer to the device-specific datasheet for the module identification number of its BECU.

22.2.12 BECU NMI Enable Register - (BECUn_NMIEN)

This register can be used to disable the propagation of the NMI status to the Interrupt Controller.

BECU NMI Enable Register (BECUn_NMIEN)

Figure 22-13. BECU NMI Enable Register (BECUn_NMIEN)

BECUn_NMIEN															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	NMIEN
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 22-13. BECU NMI Enable Register (BECUn_NMIEN) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	NMIEN	Non-Maskable Interrupt Enable This is a one-time writable bit, i.e. only the first write access after a reset can clear this bit. Once cleared, it can be set only by a reset. Any subsequent write access after the initial write access triggers an error response. '0': NMI is disabled. In other words, the NMI status is not propa- gated to the Interrupt Controller '1': NMI is enabled. In other words, the NMI status is propagated to the Interrupt Controller

22.3 Operation of BECU

This chapter describes the operation of the BECU.

Bus interface

The bus masters in the MCU can access the BECU module through its (e)RBUS slave interface. The address decoder decodes the (e)RBUS address bus. If the access is for a register, the same is routed to the CSR. Register accesses are masked by the peripheral protection logic, which are based on the peripheral protection policies.

Memory Map/Address Space

Each PERIn_RBUS section contains its own BECUn instance, which is assigned to the address space of the bus section. Please see the Memory Map of the specific device datasheet for details about assigned addresses.

Transaction pipeline

All transactions on the (e)RBUS are continuously sampled by the BECU and pushed into its internal transaction pipeline if the selected slave does not insert any wait states on the bus.

Control and Status

Registers The operation of the BECU can be controlled and monitored through its CSRs, details of which can be found in 22.2 BECU registers.

22.4 Notes on using the BECU

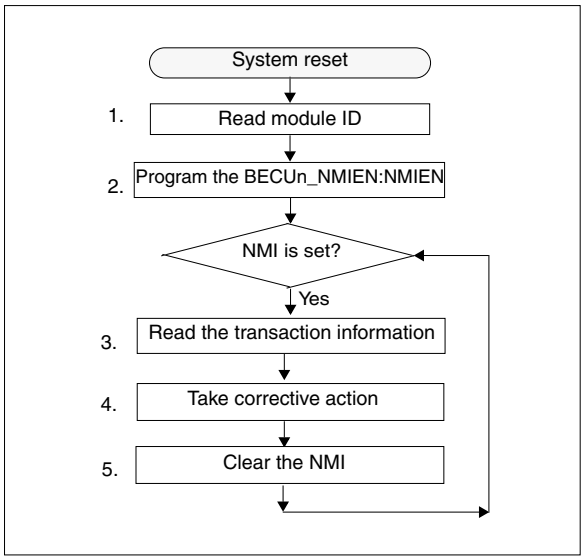
This section is the 'programmer's guide', which lists the usage notes for programming the BECU module. It is recommended that programmers read these guidelines before programming the BECU module.

General usage notes

- Reserved bits return undefined values. The software programs shall be independent of the values read from the reserved register bits
- The BECU supports storage of information of only one bus access at a time. Therefore, once an error is captured (and the BECUn_CTRL:NMI bit is set), further 'Error Collect' triggers are ignored until the BECUn_CTRL:NMI bit is cleared. This implies that any errors occurring on the bus while the BECUn_CTRL:NMI is set are simply ignored by the BECU. Software developers must therefore try to keep the Interrupt Service Routine (ISR) size small so that the NMI service request of the BECU does not remain unattended for long

Steps in programming the BECU module

Figure 22-14. Steps in programming the BECU module



The following steps describe the programming of the BECU module:

1. After the system reset, the software shall detect the BESU module ID number by reading the BECUn_MID register. This would help it in identifying the attributes and capabilities supported by the BECU module.
2. By default, BECU propagates the NMI service request to the CPU through the Interrupt Controller. If polling mode is desired, the software can reset the BECUn_NMIEN:NMIEN bit to '0'. It should be noted that the BECUn_NMIEN:NMIEN can be written only once after reset. Subsequent write accesses to this bit have no visible impact on the state of this bit.
3. When the NMI is triggered, (or if in polling mode the software detects that the BECUn_CTRL:NMI status flag is set during its polling cycle), the CPU is invoked, which would read the status information collected and stored by the BECU in its CSR.
4. The software diagnoses the information about the (e)RBUS transaction for the error response signalled and initiates corrective action (if any).
5. While the BECUn_CTRL:NMI flag is set, the status registers that store the information collected about the erroneous transaction from the (e)RBUS are locked for further updates. This implies that any further bus errors would be ignored unless the BECUn_CTRL:NMI flag is cleared by the software. Once the software has processed the information from these status registers, it shall clear the BECUn_CTRL:NMI flag by writing '1' to the BECUn_CTRL:NMICL bit. Clearing the BECUn_CTRL:NMI flag ensures that the BECU is again ready to capture bus errors.

23. RETENTIONRAM

This chapter explains the function and operation of the RETENTIONRAM module.

23.1 Outline of the RETENTIONRAM

The RETENTIONRAM module connects to synchronous static RAM. It has an ECC logic which can generate an interrupt during single-bit error detection in RETENTIONRAM read data. The ECC logic is made testable with the option of injecting bit errors into RETENTIONRAM read data. This section describes the features and the block diagram of the RETENTIONRAM module.

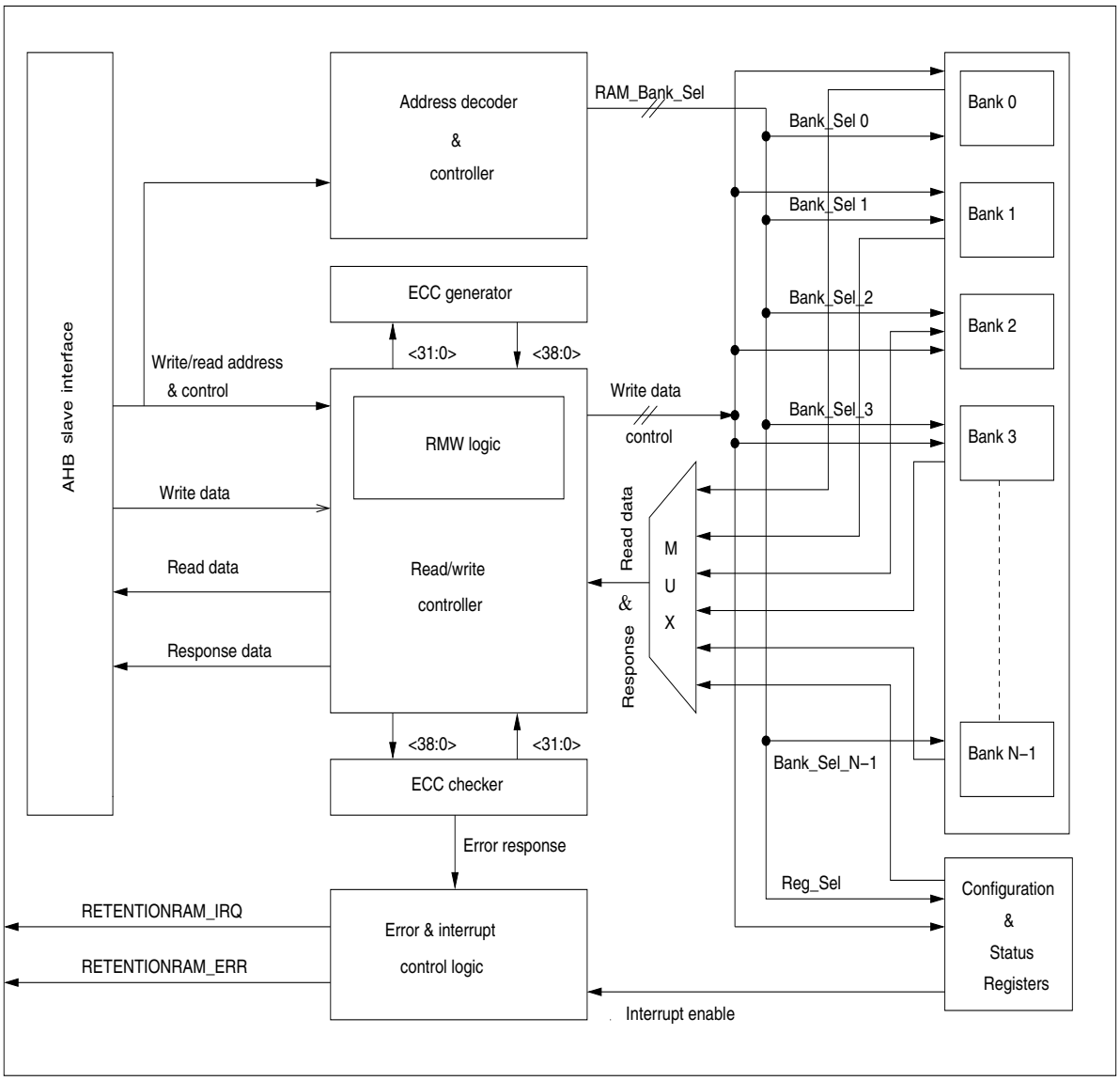
Features of the RETENTIONRAM

The RETENTIONRAM module connects to synchronous static RAM. Other features of this module are:

- Configurable ECC support for RAM
- It generates an interrupt on a single-bit error detection when interrupt generation is enabled
- It generates a bus error response on non-correctable ECC errors
- It supports error injection by corrupting the data bits and the ECC check bits read from the RETENTIONRAM to test the ECC functionality
- The RETENTIONRAM resides in a separate power domain which can be kept ON even when the processor is powered down in power save state. This provides the possibility to save vital data in the RAM before entering power-down state
- Individual memory banks can be disabled for maximum power reduction. See description of BSU registers BSU7_PEN7 and BSU7_PEN8 on how to enable or disable individual memory banks

Block diagram of the RETENTIONRAM

Figure 23-1. Block diagram of the RETENTIONRAM



23.2 RETENTIONRAM registers

All registers in the RETENTIONRAM module are explained in this section. RETENTIONRAM can be programmed and monitored through its Configuration and Status Registers (CSRs).

Registers of the RETENTIONRAM

The following registers are available for RETENTIONRAM:

- RETENTIONRAM Unlock Register (RRCFG_UNLOCKR)
- RETENTIONRAM Configuration and Status Register (RRCFG_CSR)
- RETENTIONRAM ECC Error Address Number Register (RRCFG_EAN)
- RETENTIONRAM Error Mask Register 0 (RRCFG_ERRMSKR0)
- RETENTIONRAM Error Mask Register 1 (RRCFG_ERRMSKR1)
- RETENTIONRAM ECC Enable Register (RRCFG_ECCEN)

Memory layout of the RETENTIONRAM registers

The RETENTIONRAM uses 1 KB of the CPU's address space for mapping the Control and Status Registers (CSRs). All RETENTIONRAM registers are shown in Figure 23-1. The RETENTIONRAM registers (except for RRCFG_UNLOCKR register) can be accessed by the CPU with 8-, 16- or 32-bit accesses.

Table 23-1. Memory layout of the RETENTIONRAM registers

Offset	+3	+2	+1	+0
0x00000000	RRCFG_UNLOCKR 00000000 00000000 00000000 00000000			
0x00000004	RRCFG_CSR 00001111 00000000 00000010 00000000			
0x00000008	RRCFG_EAN 00000000 00000000 00000000 00000000			
0x0000000C	RRCFG_ERRMSKR0 00000000 00000000 00000000 00000000			
0x00000010	RRCFG_ERRMSKR1 00000000 00000000 00000000 00000000			
0x00000014	RRCFG_ECCEN 00000000 00000000 00000000 00000001			

23.2.1 RETENTIONRAM Unlock Register (RRCFG_UNLOCKR)

The RETENTIONRAM Unlock Register can be used for locking/unlocking RETENTIONRAM CSRs for further write accesses.

RETENTIONRAM Unlock Register (RRCFG_UNLOCKR)

Figure 23-2. RETENTIONRAM Unlock Register (RRCFG_UNLOCKR)

RRCFG_UNLOCKR																															
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**
KEY[31]	KEY[30]	KEY[29]	KEY[28]	KEY[27]	KEY[26]	KEY[25]	KEY[24]	KEY[23]	KEY[22]	KEY[21]	KEY[20]	KEY[19]	KEY[18]	KEY[17]	KEY[16]	KEY[15]	KEY[14]	KEY[13]	KEY[12]	KEY[11]	KEY[10]	KEY[9]	KEY[8]	KEY[7]	KEY[6]	KEY[5]	KEY[4]	KEY[3]	KEY[2]	KEY[1]	KEY[0]
R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP	R0WP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-2. RETENTIONRAM Unlock Register (RRCFG_UNLOCKR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	KEY	Unlock Register This register locks or unlocks all configuration registers of RETENTIONRAM, to avoid accidental overwriting by the software. Writing the lock key 0xDECCB10C to this register locks all accesses to the RETENTIONRAM Configuration and Status Registers (CSRs). The unlock key 0xACC5DECC unlocks all accesses to the RETENTIONRAM CSRs. The correct key for locking or unlocking the RETENTIONRAM CSRs can be written in privileged mode only. Accessing the RETENTIONRAM CSRs while they are locked or writing an invalid lock key or unlock key to this register triggers a bus error response. Only 32-bit accesses are supported by this register. An 8- or 16-bit read or write access to this register triggers a bus error response. The following registers are protected by this lock/unlock feature: • RRCFG_CSR • RRCFG_EAN • RRCFG_ERRMSKR0 • RRCFG_ERRMSKR1 • RRCFG_ECCEN

23.2.2 RETENTIONRAM Configuration and Status Register (RRCFG_CSR)

The RETENTIONRAM CSR can be used for configuration of the number of wait states required for the read/write access to the RAM. It can also be used for enabling/clearing the ECC correctable error interrupt flag and for enabling/disabling the

ECC error calculation. 8-, 16- and 32-bit accesses are allowed to this register.

This register also gives the lock/unlock status of the RETENTIONRAM register set and the status of ECC correctable error interrupt flag.

RETENTIONRAM Configuration and Status Register (RRCFG_CSR)

Figure 23-3. RETENTIONRAM Configuration and Status Register (RRCFG_CSR)

RRCFG_CSR																															
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**
read0	read0	read0	read0	WAWC1[1]	WAWC1[0]	RAWC1[1]	RAWC1[0]	read0	read0	read0	read0	read0	read0	read0	CEIC	read0	read0	read0	read0	read0	read0	LCK	CEIF	read0	read0	read0	read0	read0	read0	read0	CEIEN
R0	R0	R0	R0	RWP	RWP	RWP	RWP	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R	R	R0	R0	R0	R0	R0	R0	R0	RWP
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 23-3. RETENTIONRAM Configuration and Status Register (RRCFG_CSR) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:26]	WAWC1	RAM Write Access Wait Cycle Configuration This register field allows configuration of the number of wait cycles inserted during the RAM write accesses. '00': RAM write accesses do not need any wait states '01': RAM write accesses need 1 wait state '10': RAM write accesses need 2 wait states '11': RAM write accesses need 3 wait states The wait cycles are inserted on the RETENTIONRAM slave interface for read accesses to RAM.
[25:24]	RAWC1	RAM Read Access Wait Cycle Configuration This register field allows configuration of the number of wait cycles inserted during the RAM read accesses. '00': RAM read accesses do not need any wait states '01': RAM read accesses need 1 wait state '10': RAM read accesses need 2 wait states '11': RAM read accesses need 3 wait states The wait cycles are inserted on the RETENTIONRAM slave interface for write accesses to RAM.
[23:17]	read0	-
[16]	CEIC	ECC Correctable Error Interrupt Clear '0': No effect '1': Clears the ECC correctable error interrupt flag (RRCFG_CSR:CEIF) Read access to this register always returns a '0'.
[15:10]	read0	-

Table 23-3. RETENTIONRAM Configuration and Status Register (RRCFG_CSR) bits

Bit Position	Bit Field Name	Bit Description
[9]	LCK	Lock This bit indicates the lock status of the RETENTIONRAM CSRs. '0': The configuration and status registers of the RETENTION-RAM are unlocked '1': The configuration and status registers of the RETENTION-RAM are locked
[8]	CEIF	ECC Correctable Error Interrupt Flag This bit is set when a correctable ECC error occurs on any read access to the RAM. This flag can be cleared using the RRCFG_CSR:CEIC bit.
[7:1]	read0	-
[0]	CEIEN	ECC Correctable Error Interrupt Enable This bit can be used to enable or disable the ECC correctable error interrupt. '0': ECC correctable error interrupt disabled '1': ECC correctable error interrupt enabled

23.2.3 RETENTIONRAM ECC Error Address Number Register (RRCFG_EAN)

The RETENTIONRAM ECC Error Address Number Register gives the absolute 32-bit address of the last detected single-bit error (i.e. correctable ECC error).

RETENTIONRAM ECC Error Address Number Register (RRCFG_EAN)

Figure 23-4. RETENTIONRAM ECC Error Address Number Register (RRCFG_EAN)

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

Table 23-4. RETENTIONRAM ECC Error Address Number Register (RRCFG_EAN) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ADR	ECC Address Number This is the address of the last correctable ECC error. When a correctable ECC error is detected by the ECC logic in the slave interface of RETENTIONRAM, the address of the RAM location where the error was detected is stored in this register. The address stored in this register is the absolute 32-bit address of the location where the error was detected.

23.2.4 RETENTIONRAM Error Mask Register 0 (RRCFG_ERRMSKR0)

The RETENTIONRAM Error Mask Register 0 can be used to configure bits [31:0] of the mask used for ECC error injection. These register bits shall be set if and only if ECC error injection is intended. By setting a bit to '1', an ECC error is injected into the corresponding RAM data bits to test the ECC logic. For normal operation of the ECC logic, the mask bits shall be '0'.

RETENTIONRAM Error Mask Register 0 (RRCFG_ERRMSKR0)

Figure 23-5. RETENTIONRAM Error Mask Register 0 (RRCFG_ERRMSKR0)

RRCFG_ERRMSKR0																															
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**
MSK[31]	MSK[30]	MSK[29]	MSK[28]	MSK[27]	MSK[26]	MSK[25]	MSK[24]	MSK[23]	MSK[22]	MSK[21]	MSK[20]	MSK[19]	MSK[18]	MSK[17]	MSK[16]	MSK[15]	MSK[14]	MSK[13]	MSK[12]	MSK[11]	MSK[10]	MSK[9]	MSK[8]	MSK[7]	MSK[6]	MSK[5]	MSK[4]	MSK[3]	MSK[2]	MSK[1]	MSK[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-5. RETENTIONRAM Error Mask Register 0 (RRCFG_ERRMSKR0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MSK	ECC Error Mask [31:0] This register contains bits [31:0] of the bit mask for ECC error injection. It decides which bits of SRAM read data is corrupted for injecting errors.

23.2.5 RETENTIONRAM Error Mask Register 1 (RRCFG_ERRMSKR1)

The RETENTIONRAM Error Mask Register 1 can be used to configure bits [38:32] of the mask used for ECC error injection. These register bits shall be set if and only if ECC error injection is intended. By setting a bit to '1' an ECC error is injected into the corresponding RAM data bits to test the ECC logic. For normal operation of the ECC logic, the mask bits shall be '0'.

RETENTIONRAM Error Mask Register 1 (RRCFG_ERRMSKR1)

Figure 23-6. RETENTIONRAM Error Mask Register 1 (RRCFG_ERRMSKR1)

RRCFG_ERRMSKR1																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	MSK[6]	MSK[5]	MSK[4]	MSK[3]	MSK[2]	MSK[1]	MSK[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-6. RETENTIONRAM Error Mask Register 1 (RRCFG_ERRMSKR1) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6:0]	MSK	ECC Error Mask [38:32] This register contains bits [38:32] of the bit mask for ECC error injection. It decides which bit of the ECC check bits are corrupted for injecting errors.

23.2.6 RETENTIONRAM ECC Enable Register (RRCFG_ECCEN)

The RETENTIONRAM ECC Enable Register can be used to disable the ECC logic of the RETENTIONRAM. By default, ECC calculation is enabled. The software can clear this bit to bypass the ECC calculation logic. This bit can be written only once. A subsequent write access to this bit triggers an error response on the bus.

RETENTIONRAM ECC Enable Register (RRCFG_ECCEN)

Figure 23-7. RETENTIONRAM ECC Enable Register (RRCFG_ECCEN)

RRCFG_ECCEN																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ECCEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 23-7. RETENTIONRAM ECC Enable Register (RRCFG_ECCEN) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	ECCEN	ECC Calculation Enable This bit can be written only once after reset. Any subsequent write access to this register triggers an error response. '0': ECC calculation is disabled '1': ECC calculation is enabled (Default) By default, ECC calculation is enabled.

23.3 Operation of the RETENTIONRAM

This chapter describes the operation of RETENTIONRAM.

ECC operation

An ECC logic enable/disable bit, RRCFG_ECCEN:ECCEN, fully enables/disables ECC logic. When this bit is enabled, the checks for ECC will be done for every read data from the RETENTIONRAM. RRCFG_CSR:CEIF is set when ECC checker logic detects a single-bit error in the RAM read data. Single-bit errors will be detected and corrected while some non-correctable errors (double bit and some multi bit errors) will be detected and an error response will be given to the bus and to the CPU directly, resulting in a data abort exception instead of NMI.

Note:

ECC logic can detect and correct all single-bit errors. It can detect all double bit errors and only some multi bit errors. RETENTIONRAM shall be initialized with write access sizes greater than or equal to 32 bits because a (39,32) Hamming code is used for ECC parity calculation. Access sizes smaller than 32 bits may result in ECC errors as RETENTIONRAM contains random data and ECC parity bits after power on.

ECC test

This feature is implemented to test the 32-bit ECC logic implemented in the RETENTIONRAM module. This is done by injecting bit flips into either the data bits or the ECC data bits, or into both ECC and data bits that are read from the RETENTIONRAM.

This is achieved by writing the required data into the configuration registers of the RETENTIONRAM when the RRCFG_CSR:LCK bit is zero. If the errors are to be introduced by corrupting the data, then 32-bit Error Mask data has to be written to the RRCFG_ERRMSKR0 register. If the errors are to be introduced by corrupting the ECC check bits then 7-bit mask data has to be written to RRCFG_ERRMSKR1 register.

Bus error response generation

The RETENTIONRAM generates a bus error response under the following conditions:

- When access to the RETENTIONRAM configuration register is attempted before the unlock condition and writing a value other than the correct unlock or lock value to the RRCFG_UNLOCKR register
- Writing to the RRCFG_UNLOCKR register with a data width other than 32 bits
- Non-privileged write access to the RETENTIONRAM configuration registers
- When a read/write operation is done on a masked memory bank
- If write access is made to read-only registers
- Writing to the RRCFG_ECCEN:ECCEN bit more than once.

Interrupt generation

The RETENTIONRAM generates an interrupt under the following conditions:

RRCFG_CSR:CEIF is set when ECC checker logic detects a single-bit error in the RAM read data. A single-bit ECC error interrupt is generated if the single error interrupt enable bit (RRCFG_CSR:CEIEN) is set.

The RETENTIONRAM module also stores the address of the RAM location where the single-bit error has occurred in the RRCFG_EAN register.

Address decoder and controller

The address decoder and controller block generate accesses to memory as well as configuration/ status registers.

An Advanced High-Performance Bus (AHB) error response will be generated with an invalid access to either the memory or configuration/status registers.

Wait states

Wait states can be configured for this module by writing to the RRCFG_CSR:WAWC1 and RRCFG_CSR:RAWC1 register fields through the AHB slave interface in privileged mode and only when the RRCFG_CSR:LCK bit is zero. A minimum of zero and a maximum of three wait states can be configured. The default number of wait states configured is three.

Before using the RETENTIONRAM module the number of wait states must be configured. Once configured, this number applies until it is changed by writing to the configuration registers.

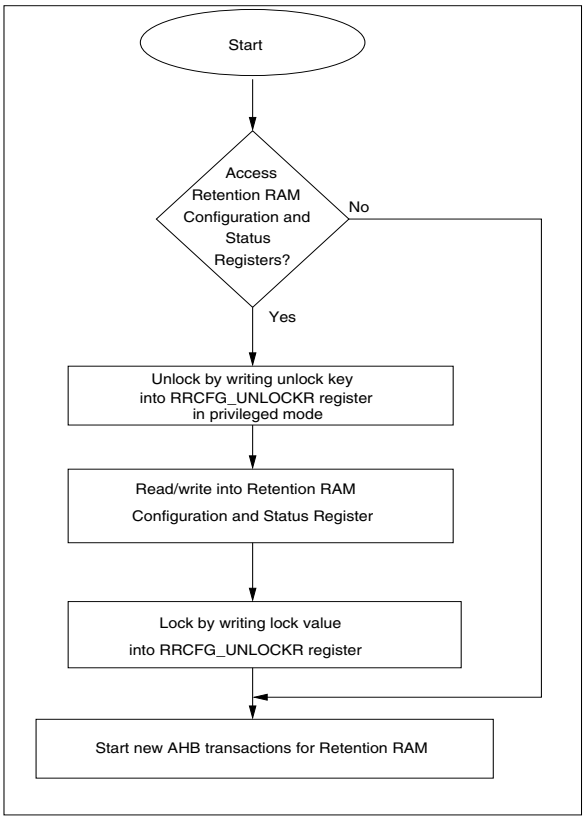
Note: The supported wait state settings can be found in the device-specific data sheet.

23.4 Notes on using the RETENTIONRAM

The following flowchart shows the sequence of steps to be followed to configure the RETENTIONRAM Configuration Registers.

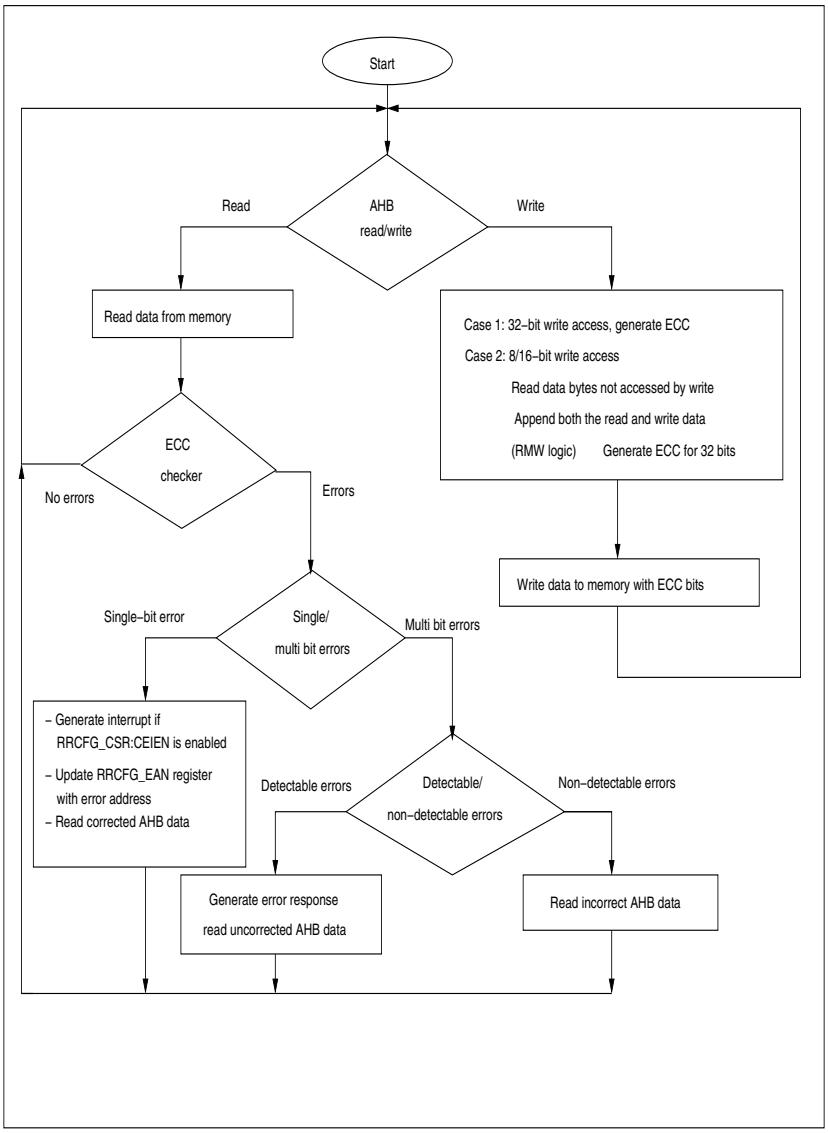
Typical RAM interface configuration flow

Figure 23-8. Typical RAM interface configuration flow



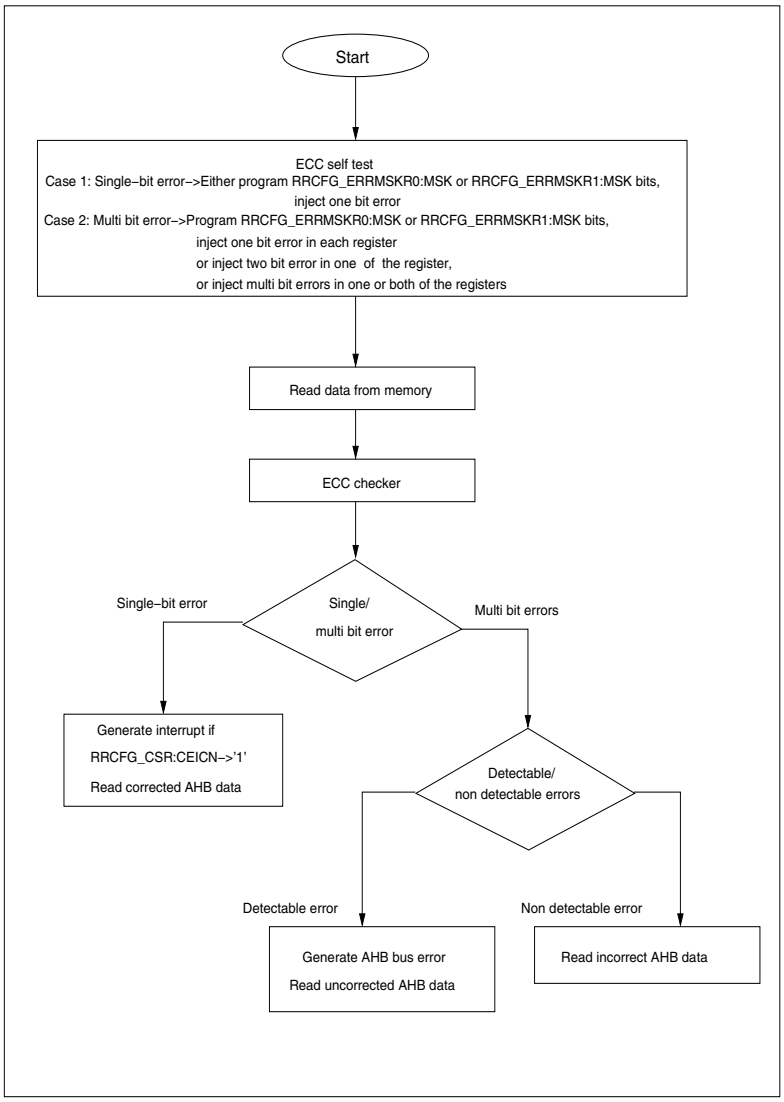
Typical flow of ECC logic operation (memory read/write)

Figure 23-9. Typical flow of ECC logic operation



Typical ECC self test flow

Figure 23-10. Typical ECC self-test flow



24. External Interrupts

This chapter explains the function and operation of the External Interrupts module.

24.1 Outline of the External Interrupts module

This section describes the features and block diagram of the External Interrupts module.

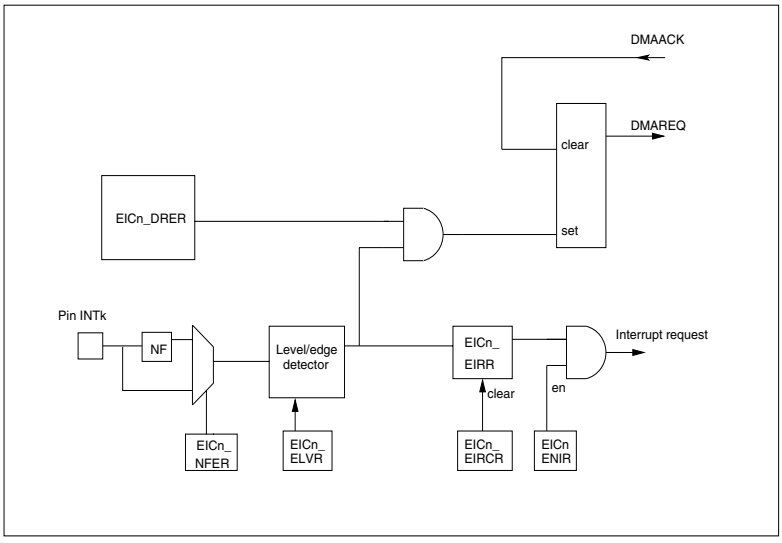
Features of the External Interrupts module

The External Interrupts module detects a signal input to an External Interrupt pin and generates an interrupt request.

- For an External Interrupt request, five levels are recognised: High ('H'), Low ('L'), rising edge, falling edge and any edge (rising or falling)
- There is External Interrupt support for 32 pins
- The External Interrupts module supports a noise filter (< 25 ns filtered-typical conditions)
- It also features bypassable noise filters
- The External Interrupts module supports a Non-Maskable Interrupt
- It provides support for Direct Memory Access (DMA) for 32 channels
- It supports an External Interrupt Capture Unit (EICU)

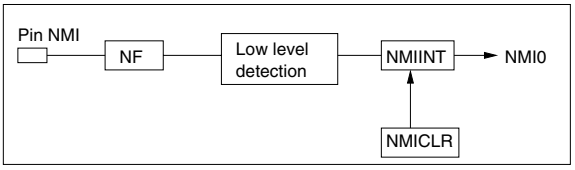
Block diagram of External Interrupts

Figure 24-1. Block diagram of External Interrupts



Block diagram of a Non-Maskable Interrupt

Figure 24-2. Block diagram of a Non-Maskable Interrupt



24.2 External Interrupts module registers

This section describes the registers of the External Interrupts module in detail.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of the External Interrupts module

The following registers are available for External Interrupts.

- External Interrupt Enable Register (EICn_ENIR)
- External Interrupt Enable Set Register (EICn_ENISR)
- External Interrupt Enable Clear Register (EICn_ENICR)
- External Interrupt Request Register (EICn_EIRR)
- External Interrupt Request Clear Register (EICn_EIRCR)
- Noise Filter Enable Register (EICn_NFER)
- Noise Filter Enable Set Register (EICn_NFESR)
- Noise Filter Enable Clear Register (EICn_NFECR)
- External Interrupt Level Register (EICn_ELVR0~3)
- Non-Maskable Interrupt Register (EICn_NMIR)
- DMA Request Enable Register (EICn_DRER)
- DMA Request Enable Set Register (EICn_DRESR)
- DMA Request Enable Clear Register (EICn_DRECR)
- DMA Request Flag Register (EICn_DRFR)

Memory layout of the External Interrupts module registers

Table 24-1. Memory layout of the External Interrupts modules registers

Offset	+3	+2	+1	+0
0x00000000	EICn_ENIR 00000000 00000000 00000000 00000000			
0x00000004	EICn_ENISR 00000000 00000000 00000000 00000000			
0x00000008	EICn_ENICR 00000000 00000000 00000000 00000000			
0x0000000C	EICn_EIRR XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000010	EICn_EIRCR 00000000 00000000 00000000 00000000			
0x00000014	EICn_NFER 00000000 00000000 00000000 00000000			
0x00000018	EICn_NFESR 00000000 00000000 00000000 00000000			
0x0000001C	EICn_NFECR 00000000 00000000 00000000 00000000			
0x00000020	EICn_ELVR0 00000000 00000000 00000000 00000000			

Table 24-1. Memory layout of the External Interrupts modules registers

Offset	+3	+2	+1	+0
0x00000024	EICn_ELVR1 00000000 00000000 00000000 00000000			
0x00000028	EICn_ELVR2 00000000 00000000 00000000 00000000			
0x0000002C	EICn_ELVR3 00000000 00000000 00000000 00000000			
0x00000030	EICn_NMIR 00000000 00000000 00000000 00000000			
0x00000034	EICn_DRER 00000000 00000000 00000000 00000000			
0x00000038	EICn_DRESR 00000000 00000000 00000000 00000000			
0x0000003C	EICn_DRECR 00000000 00000000 00000000 00000000			
0x00000040	EICn_DRFR 00000000 00000000 00000000 00000000			

24.2.1 External Interrupt Enable Register (EICn_ENIR)

This register enables the External Interrupt for the corresponding 32 External Interrupt sources.

External Interrupt Enable Register (EICn_ENIR)

Figure 24-3. External Interrupt Enable Register (EICn_ENIR)

EICn_ENIR																															
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**
EN[31]	EN[30]	EN[29]	EN[28]	EN[27]	EN[26]	EN[25]	EN[24]	EN[23]	EN[22]	EN[21]	EN[20]	EN[19]	EN[18]	EN[17]	EN[16]	EN[15]	EN[14]	EN[13]	EN[12]	EN[11]	EN[10]	EN[9]	EN[8]	EN[7]	EN[6]	EN[5]	EN[4]	EN[3]	EN[2]	EN[1]	EN[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-2. External Interrupt Enable Register (EICn_ENIR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EN	Interrupt Enable The EICn_ENIR register contains the interrupt enable bits of the External Interrupt module. '0': Interrupt disabled '1': Interrupt enabled As well as directly writing '1'/'0' to the EICn_ENIR register, it is also possible to set/clear this register by writing '1' in the EICn_ENISR/EICn_ENICR register.

Note: When an interrupt is enabled, the corresponding port's Port Input Enable Register (PPC_PCFGRijj:PIE) must also be set.

24.2.2 External Interrupt Enable Set Register (EICn_ENISR)

This register sets the corresponding EICn_ENIR bits. If a particular EICn_ENISR:ENS[31:0] bit is written to '1', the corresponding EICn_ENIR bit is set.

External Interrupt Enable Set Register (EICn_ENISR)

Figure 24-4. External Interrupt Enable Set Register (EICn_ENISR)

EICn_ENISR																															
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**
ENS[31]	ENS[30]	ENS[29]	ENS[28]	ENS[27]	ENS[26]	ENS[25]	ENS[24]	ENS[23]	ENS[22]	ENS[21]	ENS[20]	ENS[19]	ENS[18]	ENS[17]	ENS[16]	ENS[15]	ENS[14]	ENS[13]	ENS[12]	ENS[11]	ENS[10]	ENS[9]	ENS[8]	ENS[7]	ENS[6]	ENS[5]	ENS[4]	ENS[3]	ENS[2]	ENS[1]	ENS[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-3. External Interrupt Enable Set Register (EICn_ENISR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ENS	Interrupt Enable Set The EICn_ENISR register contains the interrupt set bits of the External Interrupt module. '0': No effect '1': Sets the corresponding bit in EICn_ENIR (i.e. the External Interrupt Enable Register) Reading these bits always returns '0'.

24.2.3 External Interrupt Enable Clear Register (EICn_ENICR)

This register clears the corresponding EICn_ENIR bits. If a particular EICn_ENICR:ENC[31:0] bit is written to '1', then the corresponding EICn_ENIR bit is cleared.

External Interrupt Enable Clear Register (EICn_ENICR)

Figure 24-5. External Interrupt Enable Clear Register (EICn_ENICR)

EICn_ENICR																															
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**
ENC[31]	ENC[30]	ENC[29]	ENC[28]	ENC[27]	ENC[26]	ENC[25]	ENC[24]	ENC[23]	ENC[22]	ENC[21]	ENC[20]	ENC[19]	ENC[18]	ENC[17]	ENC[16]	ENC[15]	ENC[14]	ENC[13]	ENC[12]	ENC[11]	ENC[10]	ENC[9]	ENC[8]	ENC[7]	ENC[6]	ENC[5]	ENC[4]	ENC[3]	ENC[2]	ENC[1]	ENC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-4. External Interrupt Enable Clear Register (EICn_ENICR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ENC	Interrupt Enable Clear The EICn_ENICR register contains the interrupt clear bits of the External Interrupt module. '0': No effect '1': Clears the corresponding bit in EICn_ENIR (i.e. the External Interrupt Enable Register) Reading these bits always returns '0'.

24.2.4 External Interrupt Request Register (EICn_EIRR)

This register indicates the status of the External Interrupt Request detected at the pin.

External Interrupt Request Register (EICn_EIRR)

Figure 24-6. External Interrupt Request Register (EICn_EIRR)

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

Table 24-5. External Interrupt Request Register (EICn_EIRR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ER	Interrupt Request These are read-only bits. '0': When an interrupt event is not detected '1': The EICn_EIRR register bits are set to '1' when an interrupt event is detected at the pin. This bit is cleared by writing '1' to the EICn_EIRCR register

24.2.5 External Interrupt Request Clear Register (EICn_EIRCR)

This register clears the corresponding EICn_EIRR bits. If a particular EICn_EIRCR:ERC[31:0] bit is written to '1', then the corresponding EICn_EIRR bit is cleared. If a particular level is selected and an active level is still present, a software clear will have no effect.

External Interrupt Request Clear Register (EICn_EIRCR)

Figure 24-7. External Interrupt Request Clear Register (EICn_EIRCR)

EICn_EIRCR																															
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**
ERC[31]	ERC[30]	ERC[29]	ERC[28]	ERC[27]	ERC[26]	ERC[25]	ERC[24]	ERC[23]	ERC[22]	ERC[21]	ERC[20]	ERC[19]	ERC[18]	ERC[17]	ERC[16]	ERC[15]	ERC[14]	ERC[13]	ERC[12]	ERC[11]	ERC[10]	ERC[9]	ERC[8]	ERC[7]	ERC[6]	ERC[5]	ERC[4]	ERC[3]	ERC[2]	ERC[1]	ERC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-6. External Interrupt Request Clear Register (EICn_EIRCR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	ERC	Interrupt Request Clear '0': No effect '1': Clears the corresponding interrupt request bit in the EICn_EIRR register Reading these bits always returns '0'.

24.2.6 Noise Filter Enable Register (EICn_NFER)

This register enables the noise filter for the corresponding 32 External Interrupt sources.

Noise Filter Enable Register (EICn_NFER)

Figure 24-8. Noise Filter Enable Register (EICn_NFER)

EICn_NFER																															
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**
NFE[31]	NFE[30]	NFE[29]	NFE[28]	NFE[27]	NFE[26]	NFE[25]	NFE[24]	NFE[23]	NFE[22]	NFE[21]	NFE[20]	NFE[19]	NFE[18]	NFE[17]	NFE[16]	NFE[15]	NFE[14]	NFE[13]	NFE[12]	NFE[11]	NFE[10]	NFE[9]	NFE[8]	NFE[7]	NFE[6]	NFE[5]	NFE[4]	NFE[3]	NFE[2]	NFE[1]	NFE[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-7. Noise Filter Enable Register (EICn_NFER) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NFE	Noise Filter Enable '0': Noise filter disabled '1': Noise filter enabled As well as directly writing '1'/'0' to the EICn_NFER register, it is also possible to set/clear this register by writing '1' to the EICn_NFESR/EICn_NFECR register.

24.2.7 Noise Filter Enable Set Register (EICn_NFESR)

This register is used to set the corresponding EICn_NFER bits. If a particular EICn_NFESR:NFES[31:0] bit is written to '1', then the corresponding EICn_NFER bit is set.

Noise Filter Enable Set Register (EICn_NFESR)

Figure 24-9. Noise Filter Enable Set Register (EICn_NFESR)

EICn_NFESR																															
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**
NFES[31]	NFES[30]	NFES[29]	NFES[28]	NFES[27]	NFES[26]	NFES[25]	NFES[24]	NFES[23]	NFES[22]	NFES[21]	NFES[20]	NFES[19]	NFES[18]	NFES[17]	NFES[16]	NFES[15]	NFES[14]	NFES[13]	NFES[12]	NFES[11]	NFES[10]	NFES[9]	NFES[8]	NFES[7]	NFES[6]	NFES[5]	NFES[4]	NFES[3]	NFES[2]	NFES[1]	NFES[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-8. Noise Filter Enable Set Register (EICn_NFESR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NFES	Noise Filter Enable Set The EICn_NFESR register contains the noise filter set bits of the External Interrupt module. '0': No effect '1': Sets the corresponding bit in EICn_NFER (i.e. the Noise Filter Enable Register) Reading these bits always returns '0'.

24.2.8 Noise Filter Enable Clear Register (EICn_NFECR)

This register clears the corresponding EICn_NFER bits. If a particular EICn_NFECR:NFEC[31:0] bit is written to '1', the corresponding EICn_NFER bit is cleared.

Noise Filter Enable Clear Register (EICn_NFECR)

Figure 24-10. Noise Filter Enable Clear Register (EICn_NFECR)

EICn_NFECR																															
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**
NFEC[31]	NFEC[30]	NFEC[29]	NFEC[28]	NFEC[27]	NFEC[26]	NFEC[25]	NFEC[24]	NFEC[23]	NFEC[22]	NFEC[21]	NFEC[20]	NFEC[19]	NFEC[18]	NFEC[17]	NFEC[16]	NFEC[15]	NFEC[14]	NFEC[13]	NFEC[12]	NFEC[11]	NFEC[10]	NFEC[9]	NFEC[8]	NFEC[7]	NFEC[6]	NFEC[5]	NFEC[4]	NFEC[3]	NFEC[2]	NFEC[1]	NFEC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-9. Noise Filter Enable Clear Register (EICn_NFECR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	NFEC	Noise Filter Enable Clear The EICn_NFECR register contains the interrupt clear bits of the External Interrupt module. '0': No effect '1': Clears the corresponding bit in EICn_NFER (i.e the Noise Filter Enable Register) Reading these bits always returns '0'.

24.2.9 External Interrupt Level Register (EICn_ELVR0~3)

These registers define the request events at the external pins. Each of the 32 pins is assigned three bits as described in Table 24-11. If a request is detected by the edge/ level detector, the interrupt flag is set as long as the input is at the specified level, even after the flag is reset by the software.

External Interrupt Level Register (EICn_ELVR0)

Figure 24-11. External Interrupt Level Register (EICn_ELVR0)

EICn_ELVR0																															
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**
read0	LC7	LB7	LA7	read0	LC6	LB6	LA6	read0	LC5	LB5	LA5	read0	LC4	LB4	LA4	read0	LC3	LB3	LA3	read0	LC2	LB2	LA2	read0	LC1	LB1	LA1	read0	LC0	LB0	LA0
R0	RWP	RWP	RWP	R0	RWP	RWP	RWP	R0	RWP	RWP	RWP	R0	RWP	RWP	RWP	R0	RWP	RWP	RWP	R0	RWP	RWP	RWP	R0	RWP	RWP	RWP	R0	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-10. External Interrupt Level Register (EICn_ELVR0) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30]	LC7	External Interrupt Level Register For more details refer to Table 24-11.
[29]	LB7	External Interrupt Level Register For more details refer to Table 24-11.
[28]	LA7	External Interrupt Level Register For more details refer to Table 24-11.
[27]	read0	-
[26]	LC6	External Interrupt Level Register For more details refer to Table 24-11.
[25]	LB6	External Interrupt Level Register For more details refer to Table 24-11.
[24]	LA6	External Interrupt Level Register For more details refer to Table 24-11.
[23]	read0	-
[22]	LC5	External Interrupt Level Register For more details refer to Table 24-11.
[21]	LB5	External Interrupt Level Register For more details refer to Table 24-11.

Table 24-10. External Interrupt Level Register (EICn_ELVR0) bits

Bit Position	Bit Field Name	Bit Description
[20]	LA5	External Interrupt Level Register For more details refer to Table 24-11.
[19]	read0	-
[18]	LC4	External Interrupt Level Register For more details refer to Table 24-11.
[17]	LB4	External Interrupt Level Register For more details refer to Table 24-11.
[16]	LA4	External Interrupt Level Register For more details refer to Table 24-11.
[15]	read0	-
[14]	LC3	External Interrupt Level Register For more details refer to Table 24-11.
[13]	LB3	External Interrupt Level Register For more details refer to Table 24-11.
[12]	LA3	External Interrupt Level Register For more details refer to Table 24-11.
[11]	read0	-
[10]	LC2	External Interrupt Level Register For more details refer to Table 24-11.
[9]	LB2	External Interrupt Level Register For more details refer to Table 24-11.
[8]	LA2	External Interrupt Level Register For more details refer to Table 24-11.
[7]	read0	-
[6]	LC1	External Interrupt Level Register For more details refer to Table 24-11.
[5]	LB1	External Interrupt Level Register For more details refer to Table 24-11.
[4]	LA1	External Interrupt Level Register For more details refer to Table 24-11.
[3]	read0	-
[2]	LC0	External Interrupt Level Register For more details refer to Table 24-11.
[1]	LB0	External Interrupt Level Register For more details refer to Table 24-11.
[0]	LA0	External Interrupt Level Register For more details refer to Table 24-11.

Note: EICn_ELVR0 provides interrupt request levels for external Interrupt sources 0 to 7. Because one register accommodates eight interrupts, four such registers (EICn_ELVR0 to EICn_ELVR3) are required to cover all 32 external Interrupt sources. EICn_ELVR1 provides interrupt request levels for external interrupt sources from 8 to 15; EICn_ELVR2 for external interrupt sources from 16 to 23; and EICn_ELVR3 for external Interrupt sources from 24 to 31.

Table 24-11. Interrupt request detection factor for external interrupts

EICn_ELVRm:LCk	EICn_ELVRm:LBk	EICn_ELVRm:LAk	Interrupt request detection factor for INTk
'0'	'0'	'0'	'L' level pin input
'0'	'0'	'1'	'H' level pin input
'0'	'1'	'0'	Rising edge pin input
'0'	'1'	'1'	Falling edge pin input
'1'	x	x	Any edge i.e (rising or falling edge)

Note:

x in the table is don’t care.

k = 0..31

m = int(k/8)

24.2.10 Non-Maskable Interrupt Register (EICn_NMIR)

This register contains two bits: EICn_NMIR:NMIINT indicates the status of the Non-Maskable Interrupt and EICn_NMIR:NMICLR is used to clear EICn_NMIR:NMIINT.

Non-Maskable Interrupt Register (EICn_NMIR)

Figure 24-12. Non-Maskable Interrupt Register (EICn_NMIR)

EICn_NMIR																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	NMICLR	read0	read0	read0	read0	read0	read0	read0	NMIINT
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0WP1	R0	R0	R0	R0	R0	R0	R0	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-12. Non-Maskable Interrupt Register (EICn_NMIR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	read0	-
[15:9]	read0	-
[8]	NMICLR	Non-Maskable Interrupt Clear '0': No effect '1': Clear the EICn_NMIR:NMIINT bit Reading this bit always returns '0'.
[7:1]	read0	-
[0]	NMIINT	Non-Maskable Interrupt This is a read-only bit. '0': A Non-Maskable Interrupt event has not been detected '1': A Non-Maskable Interrupt event has been detected. The EICn_NMIR:NMIINT register bit is cleared by setting the EICn_NMIR:NMICLR bit to '1'

24.2.11 DMA Request Enable Register (EICn_DRER)

This register enables the DMA for the External Interrupts module.

DMA Request Enable Register (EICn_DRER)

Figure 24-13. DMA Request Enable Register (EICn_DRER)

EICn_DRER																															
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**
DRE[31]	DRE[30]	DRE[29]	DRE[28]	DRE[27]	DRE[26]	DRE[25]	DRE[24]	DRE[23]	DRE[22]	DRE[21]	DRE[20]	DRE[19]	DRE[18]	DRE[17]	DRE[16]	DRE[15]	DRE[14]	DRE[13]	DRE[12]	DRE[11]	DRE[10]	DRE[9]	DRE[8]	DRE[7]	DRE[6]	DRE[5]	DRE[4]	DRE[3]	DRE[2]	DRE[1]	DRE[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-13. DMA Request Enable Register (EICn_DRER) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DRE	DMA Enable The EICn_DRER register contains the DMA enable bits. '0': DMA disabled '1': DMA enabled As well as directly writing '1'/'0' to the EICn_DRER register, it is also possible to set/clear this register by writing '1' to the EICn_DRESR/EICn_DRECR register

24.2.12 DMA Request Enable Set Register (EICn_DRESR)

This register sets the corresponding EICn_DRER bits. If a particular EICn_DRESR:DRES[31:0] bit is written to '1', the corresponding EICn_DRER bit is set.

DMA Request Enable Set Register (EICn_DRESR)

Figure 24-14. DMA Request Enable Set Register (EICn_DRESR)

EICn_DRESR																															
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**
DRES[31]	DRES[30]	DRES[29]	DRES[28]	DRES[27]	DRES[26]	DRES[25]	DRES[24]	DRES[23]	DRES[22]	DRES[21]	DRES[20]	DRES[19]	DRES[18]	DRES[17]	DRES[16]	DRES[15]	DRES[14]	DRES[13]	DRES[12]	DRES[11]	DRES[10]	DRES[9]	DRES[8]	DRES[7]	DRES[6]	DRES[5]	DRES[4]	DRES[3]	DRES[2]	DRES[1]	DRES[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-14. DMA Request Enable Set Register (EICn_DRESR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DRES	DMA Request Enable Set The EICn_DRESR register contains the set bits for the DMA. '0': No effect '1': Sets the corresponding bit in EICn_DRER (i.e the DMA Enable Register) Reading these bits always returns '0'.

24.2.13 DMA Request Enable Clear Register (EICn_DRECR)

This register clears the corresponding EICn_DRECR bits. If a particular EICn_DRECR:DREC[31:0] bit is written to '1', the corresponding EICn_DRER bit is cleared.

DMA Request Enable Clear Register (EICn_DRECR)

Figure 24-15. DMA Request Enable Clear Register (EICn_DRECR)

EICn_DRECR																															
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**
DREC[31]	DREC[30]	DREC[29]	DREC[28]	DREC[27]	DREC[26]	DREC[25]	DREC[24]	DREC[23]	DREC[22]	DREC[21]	DREC[20]	DREC[19]	DREC[18]	DREC[17]	DREC[16]	DREC[15]	DREC[14]	DREC[13]	DREC[12]	DREC[11]	DREC[10]	DREC[9]	DREC[8]	DREC[7]	DREC[6]	DREC[5]	DREC[4]	DREC[3]	DREC[2]	DREC[1]	DREC[0]
R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1	R0WP1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-15. DMA Request Enable Clear Register (EICn_DRECR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DREC	DMA Request Enable Clear The EICn_DRECR register contains the clear bits for the DMA. '0': No effect '1': Clears the corresponding bit in EICn_DRER (i.e the DMA Enable Register) Reading these bits always returns '0'.

24.2.14 DMA Request Flag Register (EICn_DRFR)

This register indicates the status of the DMA request.

DMA Request Flag Register (EICn_DRFR)

Figure 24-16. DMA Request Flag Register (EICn_DRFR)

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

Table 24-16. DMA Request Flag Register (EICn_DRFR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DRF	DMA Flag This is a read-only bit. '0': The EICn_DRFR register bits are cleared by the external DMA ACK pins '1': The EICn_DRFR register bits are set to '1' when EICn_DRER is enabled and an interrupt event is detected at the pin

24.3 Operation of the External Interrupts module

The section describes the operation of the External Interrupts module in detail.

Conditions on the behavior of an external circuit that uses DMA

An external circuit which uses DMA must be able to deactivate the request signal when the requested DMA transfer has been performed. Once the DMA transfers are completed, the DMA flag and DMAREQ are cleared with DMAACK. DMAACK has priority over DMAREQ.

Clearing the interrupt flag

- When it is used as an External Interrupt the interrupt flag EICn_EIRR:ER must be cleared within the interrupt service routine. Otherwise the same service is performed again after the completion of the first interrupt service
- When level detection is specified as the event input, the interrupt flag is set again even after it is cleared as long as the active level is kept at the input pin. In this case, the external cause of the request should be cleared or the interrupt enable bit cleared. A software clear has priority over a hardware set

Noise filter

The noise filter removes the noise from the signal on the INTk pin. Typically, signals with a width of less than 25 ns are filtered.

External Interrupts request level

- When edge detection is specified as the event input, the pulse of the input signal must have a minimum width to be recognized as an input edge and should not be filtered by the noise filter. For the minimum pulse width length refer to the device specific datasheet
- When level detection is specified as the event input, the interrupt flag maintains its active status once the specified level is input, even after the input signal changes to the inactive level as shown in Figure 24-18. In order to clear the request, the interrupt flag must be cleared

To change the interrupt level or noise filter, the software should use the following sequence to avoid any false triggering:

1. Disable interrupt
2. Change level
3. Clear interrupt
4. Enable interrupt

Figure 24-17. Clearing interrupt cause register upon level set

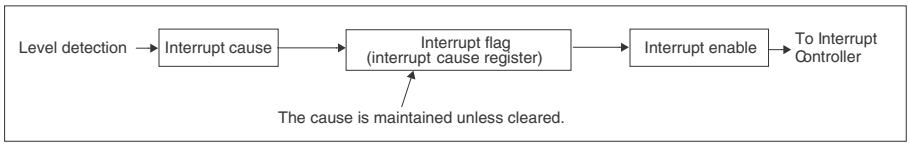


Figure 24-18. Interrupt cause and interrupt request to the interrupt controller while interrupts are enabled

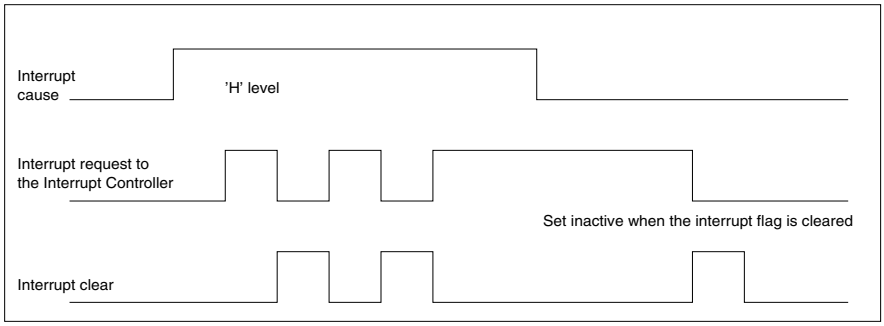
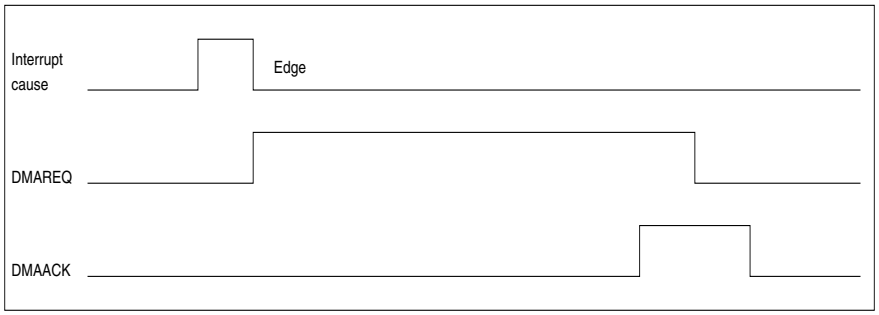


Figure 24-19. Interrupt cause and DMAREQ to DMA while DMA is enabled

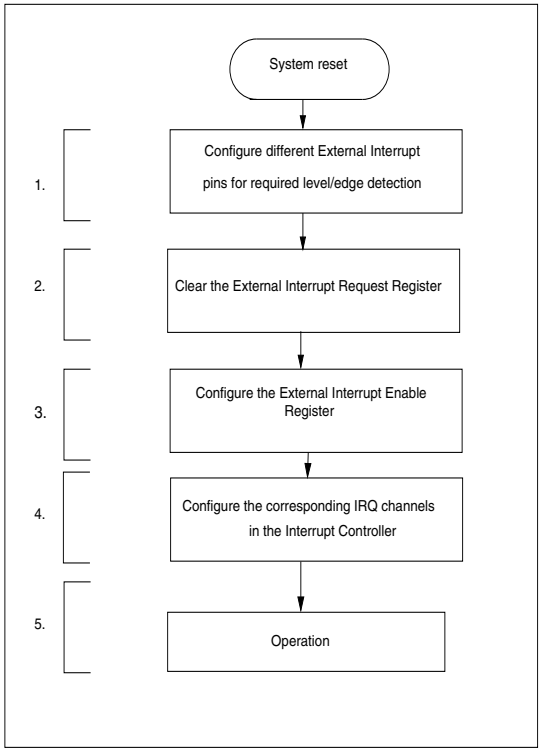


24.4 Notes on using the External Interrupts module

The following section describes the usage of the External Interrupts module.

Steps in programming the External Interrupts module

Figure 24-20. Steps in programming the External Interrupts module



The general steps a programmer should follow when using the External Interrupts modules are as follows:

1. After the system reset, configure the different External Interrupt pins for the required level/edge detection.
2. Clear the External Interrupt Request Register.
3. Configure the External Interrupt Enable Register.
4. Configure the corresponding IRQ channels in the Interrupt Controller. Refer to 17. Interrupt Controller.
5. Continue with the application program.

25. Sound Generator

This chapter explains the function and operation of the Sound Generator module.

25.1 Outline of the Sound Generator

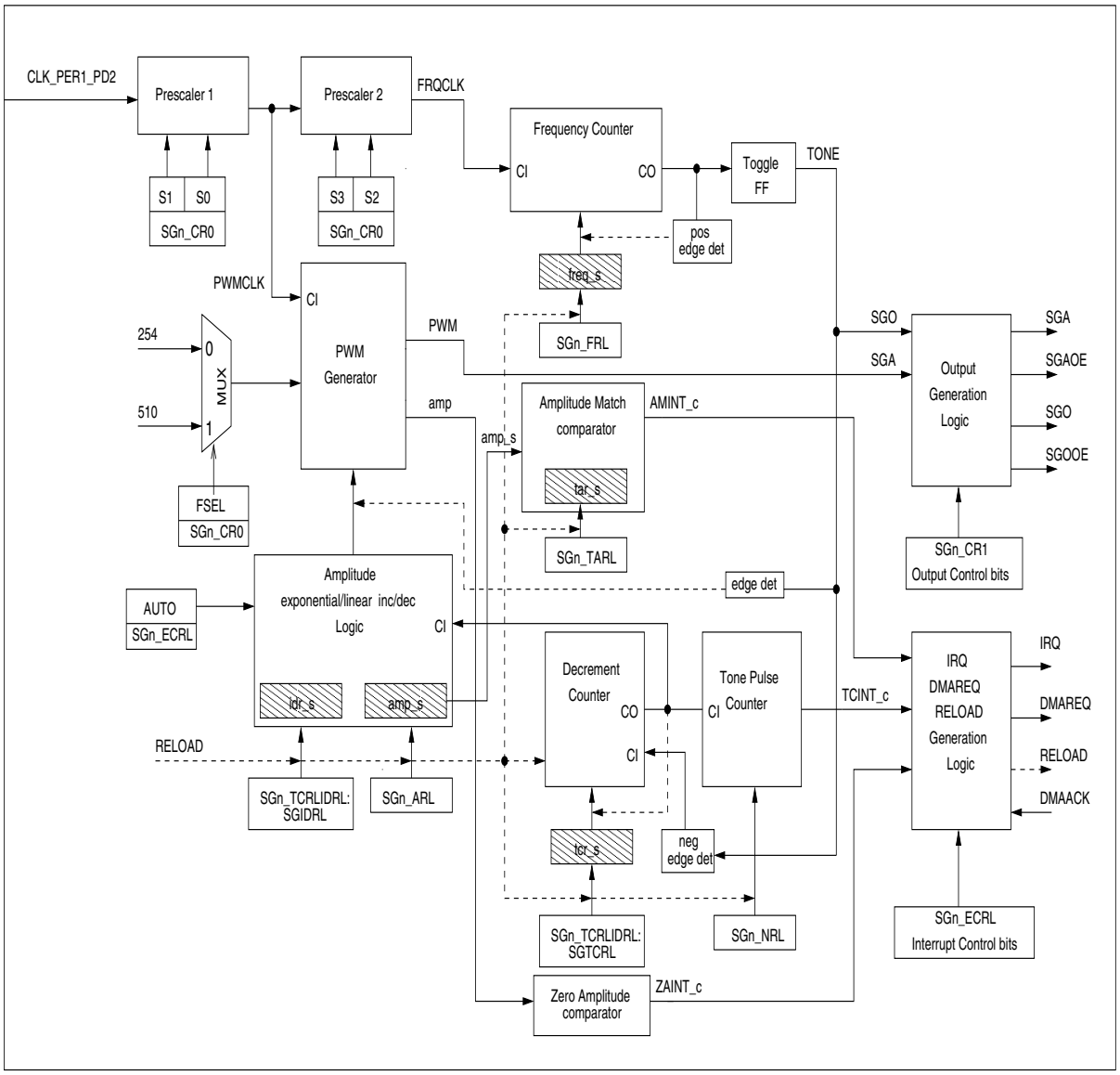
The Sound Generator IP is an advanced Pulse Width Modulation (PWM) based Sound Generator for sound/melody generation. Sound output is in the form of a square wave with programmable frequency. The amplitude of the sound output is programmable for linear or exponential increment or decrement without additional interrupt requirements. This section describes the features and the block diagram of the Sound Generator module.

Features of Sound Generator

- Flexible programming - Sound Generator programming depends on the application
- It produces sound/melody with varying frequency and amplitude for a programmable duration
- The sound output is a square wave. The resolution between 100 Hz and 6 kHz is better than at 20 Hz with an input frequency of 16 MHz
- The Pulse Width Modulation (PWM) cycle width can be programmed to be either 255 or 511 clocks, and the duty cycle (i.e. amplitude) can be programmed in the range from 0 to 100%
- The frequency and amplitude counters are driven by programmable prescalers in which the clock can be divided by 1, 2, 3 or 4
- The Sound Generator features automatic linear or exponential amplitude increment or decrement without additional interrupts
- It has functionality to start, stop and resume sound generation without having to reload the configuration
- The sound output is stopped automatically when the amplitude becomes zero
- A Dedicated sequencer is used to support optimised Direct Memory Access (DMA) data transfer
- The Sound Generator features a synchronised 'on the fly' reload of the registers on a Register Reload event condition
- A programmable interrupt, a DMA request and Register Reload event generation occur at the end of the tone pulse counter, amplitude match condition and zero amplitude condition respectively
- It features support for the Microcontroller Unit (MCU) debug mode

Block diagram of Sound Generator

Figure 25-1. Block diagram of Sound Generator



25.2 Sound Generator registers

This section describes the Sound Generators registers. All registers are byte or half-word accessible.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of Sound Generator

The following registers are available for the Sound Generator:

- Sound Generator Control Register 0 (SGn_CR0)
- Sound Generator Control Register 1 (SGn_CR1)
- Sound Generator Extended Control Reload Register (SGn_ECRL)
- Sound Generator Frequency Data Reload Register (SGn_FRL)
- Sound Generator Amplitude Status Register (SGn_AR)
- Sound Generator Amplitude Data Reload Register (SGn_ARL)
- Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL)
- Sound Generator Target Amplitude Data Reload Register (SGn_TARL)
- Sound Generator Tone Output Number Reload Register (SGn_NRL)
- Sound Generator DMA Transfer Update Enable Register (SGn_DER)
- Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR)

Memory layout of Sound Generator registers

Table 25-1. Memory layout of Sound Generator registers

Offset	+1	+0
0x00000000	SGn_CR0 00000000 00000000	
0x00000002	reserved XXXXXXXX	SGn_CR1 00000000
0x00000004	SGn_ECRL 00000000 00000000	
0x00000006	SGn_FRL 00000000 00000000	
0x00000008	SGn_ARL 00000000 00000000	
0x0000000A	SGn_AR 00000000 00000000	
0x0000000C	SGn_TARL 00000000 00000000	
0x0000000E	SGn_TCRLIDRL 00000000 00000000	
0x00000010	reserved XXXXXXXX	SGn_NRL 00000000
0x00000012	SGn_DER 00000000 00000000	

Table 25-1. Memory layout of Sound Generator registers

Offset	+1	+0
0x00000014	SGn_DMAR 00000000 00000000	

25.2.1 Sound Generator Control Register 0 (SGn_CR0)

The Sound Generator Control Register 0 (SGn_CR0) is used to control the Sound Generator operation mode, the prescaler divide factor and the clearing of the interrupt status bits. It also controls the PWM cycle period, DMA operation and debug mode selection.

Sound Generator Control Register 0 (SGn_CR0)

Figure 25-2. Sound Generator Control Register 0 (SGn_CR0)

SGn_CR0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DBGE	FSEL	DMA	SGDADS	T[1]	T[0]	S[1]	S[0]	read0	AMICLR	TCICLR	ZAICLR	RUNNING	RESUME	STOP	START
RpWp	RpWp	RpWp	Rp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-2. Sound Generator Control Register 0 (SGn_CR0)

Bit Position	Bit Field Name	Bit Description
[15]	DBGE	Debug Mode Enable This bit is used to enable/disable the debug mode of operation. '0': Debug mode is disabled '1': Debug mode is enabled When DBGE is set to '1' and the processor is in the debug state, Sound Generator operation is stopped by masking the SGn_CR0:RUNNING bit, which in turn stops all the counters and hence the sound output. For the definition of the debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[14]	FSEL	PWM Cycle Select These bits select the PWM clock period. '0': One PWM cycle is 255 clock cycles '1': One PWM cycle is 511 clock cycles
[13]	DMA	DMA Mode Select '0': Disables the DMA mode of operation '1': DMA mode of operation selected (triggers a DMA request at the start of sound generation)

Table 25-2. Sound Generator Control Register 0 (SGn_CR0)

Bit Position	Bit Field Name	Bit Description
[12]	SGDADS	Automatic Output Stop Status Configured by setting the SGn_ECRL:SGDADR bit, this bit is set when the sound amplitude becomes '0' and is cleared when the amplitude value is non-zero. When this bit is set, it clears the SGn_CR0:RUNNING bit, thereby stopping sound output. '0': Sound amplitude is non-zero and sound outputs (SGA and SGO) are not stopped '1': Sound amplitude is zero and sound outputs (SGA and SGO) are stopped
[11:10]	T	Prescaler 2 Clock Division Select These bits select the Prescaler 2 Clock Division ratio: '00': FRQCLK = PWMCLK '01': FRQCLK = PWMCLK/2 '10': FRQCLK = PWMCLK/3 '11': FRQCLK = PWMCLK/4 The output FRQCLK of Prescaler 2 is fed into the frequency counter.
[9:8]	S	Prescaler 1 Clock Division Select These bits select the Prescaler 1 Clock Division ratio: '00': PWMCLK = CLK_PERI1_PD2 '01': PWMCLK = CLK_PERI1_PD2/2 '10': PWMCLK = CLK_PERI1_PD2/3, '11': PWMCLK = CLK_PERI1_PD2/4 The output PWMCLK of Prescaler 1 is fed into the PWM generator and Prescaler 2.
[7]	read0	-
[6]	AMICLR	Amplitude Match Interrupt Clear '0': No effect '1': Clears the SGn_CR1:AMINT bit (amplitude match interrupt status bit) Reading this bit returns '0'.
[5]	TCICLR	Tone Pulse Count Interrupt Clear '0': No effect '1': Clears the SGn_CR1:TCINT bit (tone count interrupt status bit) Reading this bit returns '0'.
[4]	ZAICLR	Zero Amplitude Interrupt Clear '0': No effect '1': Clears the SGn_CR1:ZAINT bit (zero amplitude interrupt status bit) Reading this bit returns '0'.

Table 25-2. Sound Generator Control Register 0 (SGn_CR0)

Bit Position	Bit Field Name	Bit Description
[3]	RUNNING	Sound Generation Status Indicates the status of the Sound Generator operation. '0': Sound Generator is stopped '1': Sound Generator is running This bit is set by writing '1' to the SGn_CR0:START or SGn_CR0:RESUME bit. It is cleared by writing '1' to the SGn_CR0:STOP bit or if the SGn_CR0:SGDADS bit is set. Writing to this bit has no effect.
[2]	RESUME	Sound Generation Resume '0': No effect '1': Sets the SGn_CR0:RUNNING bit Reading this bit always returns '0'.
[1]	STOP	Sound Generation Stop '0': No effect '1': Resets the SGn_CR0:RUNNING bit Reading this bit always returns '0'.
[0]	START	Sound Generation Start '0': No effect '1': Sets the SGn_CR0:RUNNING bit Reading this bit always returns '0'. Upon setting the START bit, the 'Register Reload' condition is generated to load the Sound Generator register data into the respective shadow registers. Setting this bit also triggers a DMA request if the SGn_CR0:DMA bit is set.

25.2.2 Sound Generator Control Register 1 (SGn_CR1)

The Sound Generator Control Register 1 (SGn_CR1) controls sound output generation (i.e. output generation on the SGA & SGO pins). This register also holds the status of the Tone Pulse Count Interrupt, the Amplitude Match Interrupt and the Zero Amplitude Interrupt.

Sound Generator Control Register 1 (SGn_CR1)

Figure 25-3. Sound Generator Control Register 1 (SGn_CR1)

SGn_CR1							
07	06	05	04	03	02	01	00
read0	AMINT	TCINT	ZAINT	TONE	AMP	SGOOE	SGAOE
Rp0	Rp	Rp	Rp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 25-3. Sound Generator Control Register 1 (SGn_CR1) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	AMINT	Amplitude Match Interrupt This bit is set to '1' when the amplitude shadow register value matches the target amplitude value configured in the SGn_TARL register. It is cleared by writing '1' to SGn_CR0:AMI-CLR. This is a read-only bit - writing to it has no effect.
[5]	TCINT	Tone Pulse Count Interrupt This bit is set to '1' when the number of tone pulse count matches the reload time value configured in the SGn_NRL register. It is cleared by writing '1' to SGn_CR0:TCICLR. This is a read-only bit - writing to it has no effect.
[4]	ZAINT	Zero Amplitude Interrupt This bit is set to '1' when the amplitude status register (SGn_AR) value equals zero. It is cleared by writing '1' to SGn_CR0:ZAICLR This is a read-only bit - writing to it has no effect.

Table 25-3. Sound Generator Control Register 1 (SGn_CR1) bits

Bit Position	Bit Field Name	Bit Description
[3]	TONE	SGO Output Select This bit selects the Tone or Mixed-Tone PWM signal (i.e. Tone mixed with PWM) to be output on the SGO pin. '0': Mixed-Tone PWM signal is output on the SGO pin '1': Tone signal is output on the SGO pin
[2]	AMP	SGA Output Select This bit selects the PWM or Mixed-Inverted-Tone PWM signal (i.e. PWM mixed with inverted Tone) to be output on the SGA pin. '0': Mixed-Inverted-Tone PWM signal is output on the SGA pin '1': PWM signal is output on the SGA pin
[1]	SGOOE	SGO Output Enable '0': SGO output is disabled (Hi-Z) '1': SGO output is enabled
[0]	SGAOE	SGA Output Enable '0': SGA output is disabled (Hi-Z) '1': SGA output is enabled

25.2.3 Sound Generator Extended Control Reload Register (SGn_ECRL)

The Sound Generator Extended Control Reload Register (SGn_ECRL) enables or disables CPU interrupt generation, DMA requests and Register Reload event control generation. It also controls the automatic exponential/linear increment/decrement of the amplitude. This register has a shadow register and copying the reload register to the shadow register is done on a Register Reload event.

Sound Generator Extended Control Reload Register (SGn_ECRL)

Figure 25-4. Sound Generator Extended Control Reload Register (SGn_ECRL)

SGn_ECRL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
AUTO	IDS	ELS	SGDADR	read0	AMRLE	TCRLE	ZARLE	read0	AMIE	TCIE	ZAIE	read0	AMDMAE	TCDMAE	ZADMAE
RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-4. Sound Generator Extended Control Reload Register (SGn_ECRL) bits

Bit Position	Bit Field Name	Bit Description
[15]	AUTO	Automatic Amplitude Increment and Decrement Enable '0': Disables automatic amplitude increment/decrement '1': Enables automatic amplitude increment/decrement
[14]	IDS	Amplitude Increment/Decrement Select '0': Auto amplitude decrement '1': Auto amplitude increment
[13]	ELS	Exponential/Linear Select '0': Linear amplitude increment/decrement (amp = amp +/- increment/decrement value) '1': Exponential amplitude increment/decrement (amp = amp +/- (amp/32))

Table 25-4. Sound Generator Extended Control Reload Register (SGn_ECRL) bits

Bit Position	Bit Field Name	Bit Description
[12]	SGDADR	Sound Output Automatic Stop Control When set, this bit enables the automatic stopping of sound on a zero amplitude condition. '0': Disables the SGn_CR0:SGDADS bit setting to '1' on zero amplitude detection '1': Enables the SGn_CR0:SGDADS bit setting to '1' on zero amplitude detection **Note:** In the case of an Automatic Output Stop function controlled by SGn_ECRL:SGDADR, if the zero amplitude condition is reached (i.e SGn_CR0:SGDADS is set), it is necessary to either set the START bit (SGn_CR0:START) twice, or set SGn_CR0:START followed by SGn_CR0:RESUME to re-start the operation. The (first) SGn_CR0:START event copies the reload register to the shadow register and the second SGn_CR0:START or SGn_CR0:RESUME event starts sound generation.
[11]	read0	-
[10]	AMRLE	Amplitude Match Reload Enable Based on the Amplitude match interrupt condition, this bit either enables or disables the Register RELOAD event generation. '0': Disables reload operation on target amplitude match condition '1': Enables reload operation on target amplitude match condition
[9]	TCRLE	Tone Count Reload Enable Based on the Tone pulse count interrupt condition, this bit either enables or disables the Register RELOAD event generation. '0': Disables reload operation on tone pulse count condition '1': Enables reload operation on tone pulse count condition
[8]	ZARLE	Zero Amplitude Reload Enable Based on the Zero amplitude interrupt condition, this bit either enables or disables the Register RELOAD event generation. '0': Disables reload operation on zero amplitude condition '1': Enables reload operation on zero amplitude condition
[7]	read0	-
[6]	AMIE	Amplitude Match Interrupt Enable This bit determines the Amplitude match interrupt enable/disable condition for an IRQ generation (CPU). '0': Disables target amplitude match interrupt '1': Enables target amplitude match interrupt

Table 25-4. Sound Generator Extended Control Reload Register (SGn_ECRL) bits

Bit Position	Bit Field Name	Bit Description
[5]	TCIE	Tone Count Interrupt Enable This bit determines the Tone pulse count interrupt enable/disable condition for an IRQ generation (CPU). '0': Disables tone pulse count interrupt '1': Enables tone pulse count interrupt
[4]	ZAIE	Zero Amplitude Interrupt Enable This bit determines the Zero amplitude interrupt enable/disable condition for an IRQ generation (CPU). '0': Disables zero amplitude match interrupt '1': Enables zero amplitude match interrupt
[3]	read0	-
[2]	AMDMAE	Amplitude Match DMA Request Enable Amplitude match interrupt condition that either enables or disables a DMAREQ generation. '0': Disables DMA request generation on a target amplitude match condition '1': Enables DMA request generation on a target amplitude match condition
[1]	TCDMAE	Tone Pulse Count DMA Request Enable Tone pulse count interrupt condition that either enables or disables a DMAREQ generation. '0': Disables DMA request generation on the reload time condition (i.e. tone pulse count becomes '0') '1': Enables DMA request generation on the reload time condition
[0]	ZADMAE	Zero Amplitude Match DMA Request Enable Zero amplitude match interrupt condition that either enables or disables a DMAREQ generation. '0': Disables DMA request generation on zero amplitude condition '1': Enables DMA request generation on zero amplitude condition

25.2.4 Sound Generator Frequency Data Reload Register (SGn_FRL)

The Sound Generator Frequency Data Reload Register (SGn_FRL) stores the reload value of the frequency counter. The value contained in this register controls the frequency or tone output of the sound. This register has a shadow register, and copying the reload register to the shadow register is done on a Register Reload event.

Sound Generator Frequency Data Reload Register (SGn_FRL)

Figure 25-5. Sound Generator Frequency Data Reload Register (SGn_FRL)

SGn_FRL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	SGFRL[14]	SGFRL[13]	SGFRL[12]	SGFRL[11]	SGFRL[10]	SGFRL[9]	SGFRL[8]	SGFRL[7]	SGFRL[6]	SGFRL[5]	SGFRL[4]	SGFRL[3]	SGFRL[2]	SGFRL[1]	SGFRL[0]
Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-5. Sound Generator Frequency Data Reload Register (SGn_FRL) bits

Bit Position	Bit Field Name	Bit Description
[15]	read0	-
[14:0]	SGFRL	Sound Generator Frequency Data Reload Register This register stores the reload value for the frequency counter. This value controls the frequency of the sound (i.e tone signal from a toggle flip-flop). The register value is reloaded into the frequency shadow register on a Register Reload event. The shadow register value is then reloaded into the frequency counter when the counter value reaches zero. When the SGn_CR0:START bit is written '1', the value contained in the register is directly loaded into the frequency counter.

25.2.5 Sound Generator Amplitude Status Register (SGn_AR)

The Sound Generator Amplitude Status Register (SGn_AR) reflects the current value in the amplitude register in the PWM generator (i.e. sound amplitude). It is a read-only register.

Sound Generator Amplitude Status Register (SGn_AR)

Figure 25-6. Sound Generator Amplitude Status Register (SGn_AR)

SGn_AR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	SGAR[8]	SGAR[7]	SGAR[6]	SGAR[5]	SGAR[4]	SGAR[3]	SGAR[2]	SGAR[1]	SGAR[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-6. Sound Generator Amplitude Status Register (SGn_AR) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8:0]	SGAR	Sound Generator Amplitude Status Register This read-only register holds the current value of the the amplitude register in the PWM generator. Zero amplitude match comparators use this register to generate the respective interrupts.

25.2.6 Sound Generator Amplitude Data Reload Register (SGn_ARL)

The Sound Generator Amplitude Data Reload Register (SGn_ARL) stores the reload value of the amplitude data register. This register has a shadow register and copying the contents of the reload register to the shadow register is done on a Register Reload event.

Sound Generator Amplitude Data Reload Register (SGn_ARL)

Figure 25-7. Sound Generator Amplitude Data Reload Register (SGn_ARL)

SGn_ARL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	SGARL[8]	SGARL[7]	SGARL[6]	SGARL[5]	SGARL[4]	SGARL[3]	SGARL[2]	SGARL[1]	SGARL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-7. Sound Generator Amplitude Data Reload Register (SGn_ARL) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8:0]	SGARL	Sound Generator Amplitude Data Reload Register This register stores the reload value of the amplitude register. The value, which controls the sound amplitude, is reloaded into the amplitude shadow register on a Register Reload event. The reloading of the amplitude shadow register in the PWM pulse generator is done on both the positive and negative edges of the tone pulse. When SGn_CR0:START is written '1', the reload register value is directly loaded into the amplitude counter.

25.2.7 Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL)

The Sound Generator Time Cycle Data Reload Register (SGn_TCRLIDRL:SGTCRL) stores the reload value of the decrement counter. It is designed to automatically increase or decrease the value stored in the amplitude shadow register without any additional interrupts. When the automatic increase/decrease enable bit is set, each time the decrement counter counts the number of tone pulses designated by the Time Cycle Data Register, the value stored in the amplitude shadow register will increase or decrease linearly or exponentially. The Increment or Decrement Data Reload Register, SGn_TCRLIDRL:SGIDRL (bits[7:0]), stores the incremental/decremental step value for automatic linear amplitude increment/decrement. During linear amplitude increment/ decrement, the amplitude shadow register value is increased or decreased by the value contained in this register each time the decrement counter counts to the reload value. Both the SGn_TCRLIDRL:SGTCRL and SGn_TCRLIDRL:SGIDRL registers have shadow registers, and the copying of the reload registers to the shadow registers is done on a Register Reload event.

Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL)

Figure 25-8. Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL)

SGn_TCRLIDRL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
SGTCRL[7]	SGTCRL[6]	SGTCRL[5]	SGTCRL[4]	SGTCRL[3]	SGTCRL[2]	SGTCRL[1]	SGTCRL[0]	SGIDRL[7]	SGIDRL[6]	SGIDRL[5]	SGIDRL[4]	SGIDRL[3]	SGIDRL[2]	SGIDRL[1]	SGIDRL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-8. Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL)

Bit Position	Bit Field Name	Bit Description
[15:8]	SGTCRL	Sound Generator Time Cycle Data Reload Register Theses bits (or register) store the reload value for the decrement counter. The reload value is configured through software. The automatic incrementing/decrementing of the sound amplitude is done every time the decrement counter counts the number of tone pulses (from toggle flip-flop) designated by Time Cycle Data Shadow Register. **Note:** The number of pulses based on this register's value is defined as 'the register value + 1'.

Table 25-8. Sound Generator Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL)

Bit Position	Bit Field Name	Bit Description
[7:0]	SGIDRL	Sound Generator Amplitude Increment/Decrement Data Reload Register These bits (or register) store the increment/decrement value for the Amplitude register, SGn_ARL. The SGn_ARL value is incremented/decremented by the SGn_TCRLIDRL:SGIDRL value every time the decrement counter counts the number of tone pulses designated by the Time Cycle Data Shadow Register. This register is only valid in the linear increment/decrement mode of operation. (i.e. SGn_ECRL:ELS = '0'). The automatic incrementing/decrementing of the sound amplitude is controlled by the SGn_ECRL:AUTO and SGn_ECRL:IDS bits.

25.2.8 Sound Generator Target Amplitude Data Reload Register (SGn_TARL)

The Sound Generator Target Amplitude Data Reload register (SGn_TARL) is used to configure the target amplitude. The Amplitude Match Interrupt (AMINT) is generated when the value of the amplitude shadow register matches the configured target amplitude. This register has a shadow register and copying the contents of the reload register to the shadow register is done on a Register Reload event.

Sound Generator Target Amplitude Data Reload Register (SGn_TARL)

Figure 25-9. Sound Generator Target Amplitude Data Reload Register (SGn_TARL)

SGn_TARL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	SGTARL[8]	SGTARL[7]	SGTARL[6]	SGTARL[5]	SGTARL[4]	SGTARL[3]	SGTARL[2]	SGTARL[1]	SGTARL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-9. Sound Generator Target Amplitude Data Reload Register (SGn_TARL) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8:0]	SGTARL	Sound Generator Target Amplitude Data Reload Register This register stores the target amplitude value. The Amplitude Match Interrupt (AMINT) is generated when the Amplitude Shadow Register value is equal to the configured target amplitude. The value of this register is reloaded into its shadow register on a ‘Register Reload’ event.

25.2.9 Sound Generator Tone Output Number Reload Register (SGn_NRL)

The Sound Generator Tone Output Number Reload Register (SGn_NRL) stores the reload value for the tone pulse counter. The tone pulse counter is designed to reduce the frequency of interrupts. It is decremented every time the decrement counter reaches zero. When the tone pulse counter reaches zero, the Tone Pulse Count Interrupt (SGn_CR1:TCINT) is set.

Sound Generator Tone Output Number Reload Register (SGn_NRL)

Figure 25-10. Sound Generator Tone Output Number Reload Register (SGn_NRL)

SGn_NRL							
07	06	05	04	03	02	01	00
SGNRL[7]	SGNRL[6]	SGNRL[5]	SGNRL[4]	SGNRL[3]	SGNRL[2]	SGNRL[1]	SGNRL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 25-10. Sound Generator Tone Output Number Reload Register (SGn_NRL) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	SGNRL	Sound Generator Tone Output Number Reload Register This register contains the reload value for the tone pulse counter, which is designed to decrease the frequency of interrupts. The reload value for the tone pulse counter is configured through software. In hardware, the tone pulse counter is decremented with every tone pulse every time the decrement counter becomes zero. When it reaches zero, the Tone Pulse Count Interrupt bit (SGn_CR1:TCINT) is set. Reloading the register value into the tone counter is done on a register RELOAD condition. When both the Tone Output Number Register (SGn_NRL) and the Time Cycle Data Register (SGn_TCRLIDRL) are set to '0', the interrupt status bit (SGn_CR1:TCINT) is set with every tone cycle.

25.2.10 Sound Generator DMA Transfer Update Enable Register (SGn_DER)

The Sound Generator DMA Transfer Update Enable Register (SGn_DER) enables the updating of the Sound Generator Registers (SGn_ARL, SGn_FRL, SGn_NRL, SGn_TCRLIDRL, SGn_TARL, SGn_ECRL) using a DMA transfer. The values in this register enable the Sound Generator DMA finite state machine to identify the registers to be updated by a DMA transfer and updates them accordingly. The register settings also enable the state machine to recognize the number of DMA transfers, the transfer byte size and the valid byte position of the DMA Transfer Indirect Register (SGn_DMAR).

Sound Generator DMA Transfer Update Enable Register (SGn_DER)

Figure 25-11. Sound Generator DMA Transfer Update Enable Register (SGn_DER)

SGn_DER															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	NRE	TCRE	IDRE	TARE1	TARE0	ARE1	ARE0	FRE1	FRE0	CRE1	CRE0
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-11. Sound Generator DMA Transfer Update Enable Register (SGn_DER) bits

Bit Position	Bit Field Name	Bit Description
[15:11]	read0	-
[10]	NRE	DMA Update Enable for SG_NRL This bit enables the updating of the Tone Count Reload Register (SGn_NRL) through the DMA Transfer Indirect Register (SGn_DMAR) when there is a DMA transfer. '0': No update of the SGn_NRL register '1': Update of the SGn_NRL register
[9]	TCRE	DMA Update Enable for SGn_TCRLIDRL:SGTCRL This bit enables the updating of the Time Cycle Reload Register (SGn_TCRLIDRL:SGTCRL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of the SGn_TCRLIDRL:SGTCRL register '1': Update of the SGn_TCRLIDRL:SGTCRL register
[8]	IDRE	DMA Update Enable for SGn_TCRLIDRL:SGIDRL This bit enables the updating of the Increment and Decrement Register (SGn_TCRLIDRL:SGIDRL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of the SGn_TCRLIDRL:SGIDRL register '1': Update of the SGn_TCRLIDRL:SGIDRL register

Table 25-11. Sound Generator DMA Transfer Update Enable Register (SGn_DER) bits

Bit Position	Bit Field Name	Bit Description
[7]	TARE1	DMA Update Enable for SGn_TARL High byte This bit enables the updating of the Target Amplitude Register (SGn_TARL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of upper SGn_TARL register byte '1': Update of upper SGn_TARL register byte
[6]	TARE0	DMA Update Enable for SGn_TARL Low byte This bit is to enable the updating of the Target Amplitude Register (SGn_TARL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of lower SGn_TARL register byte '1': Update of lower SGn_TARL register byte
[5]	ARE1	DMA Update Enable for SGn_ARL High byte This bit enables the updating of the Amplitude Reload Register (SGn_ARL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of upper SGn_ARL register byte '1': Update of upper SGn_ARL register byte
[4]	ARE0	DMA Update Enable for SGn_ARL Low byte This bit enables the updating of the Amplitude Reload Register (SGn_ARL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of lower SGn_ARL register byte '1': Update of lower SGn_ARL register byte
[3]	FRE1	DMA Update Enable for SGn_FRL High byte This bit enables the updating of the Frequency Reload Register (SGn_FRL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of higher SGn_FRL register byte '1': Update of higher SGn_FRL register byte
[2]	FRE0	DMA Update Enable for SGn_FRL Low byte This bit enables the updating of the Frequency Reload Register (SGn_FRL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of lower SGn_FRL register byte '1': Update of lower SGn_FRL register byte

Table 25-11. Sound Generator DMA Transfer Update Enable Register (SGn_DER) bits

Bit Position	Bit Field Name	Bit Description
[1]	CRE1	DMA Update Enable for SGn_ECRL High byte This bit enables the updating of the Control Reload Register (SGn_ECRL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of upper SGn_ECRL register byte '1': Update of upper SGn_ECRL register byte
[0]	CRE0	DMA Update Enable for SGn_ECRL Low byte This bit enables the updating of the Control Reload Register (SGn_ECRL) through the DMA Transfer Indirect Data Register (SGn_DMAR) when there is a DMA transfer. '0': No update of lower SGn_ECRL register byte '1': Update of lower SGn_ECRL register byte

25.2.11 Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR)

The Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR) is used for DMA transfer to the following registers: Amplitude Data Reload Register (SGn_ARL), the Frequency Data Reload Register (SGn_FRL), the Tone Output Number Reload Register (SGn_NRL), the Time Cycle Data Reload Register & Increment or Decrement Data Reload Register (SGn_TCRLIDRL), the Target Amplitude Data Reload Register (SGn_TARL) and the Extended Control Reload Register (SGn_ECRL). The read value is always '0'. To know a value written to this register, read the value of the particular register of interest either SGn_ARL, SGn_FRL, SGn_NRL, SGn_TCRLIDRL, SGn_TARL or SGn_ECRL. Byte or half-word access to the SGn_DMAR register is allowed according to the setting of the DMA transfer update enable register. The DMA controller carries out a DMA transfer to this register after receiving a DMA request from the Sound Generator. The DMA finite state machine updates the corresponding Sound Generator registers depending on the setting in the SGn_DER register.

Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR)

Figure 25-12. Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR)

SGn_DMAR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
SGDMAR[15]	SGDMAR[14]	SGDMAR[13]	SGDMAR[12]	SGDMAR[11]	SGDMAR[10]	SGDMAR[9]	SGDMAR[8]	SGDMAR[7]	SGDMAR[6]	SGDMAR[5]	SGDMAR[4]	SGDMAR[3]	SGDMAR[2]	SGDMAR[1]	SGDMAR[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-12. Sound Generator DMA Transfer Indirect Data Register (SGn_DMAR) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	SGDMAR	DMA Transfer Indirect Data Register The DMA Transfer Indirect Register (SGn_DMAR) is an indirect register used for DMA transfer to the Sound Generator registers. The DMA controller initiates a DMA transfer to this register when it receives a DMA request from the Sound Generator. The transfer destination address of the DMA controller must be fixed to this register. A sequencer or DMA finite state machine updates the corresponding Sound Generator registers depending on the settings in the DMA Transfer Update Enable Register (SGn_DER). Since the register write combination is based on 2 byte boundary of address the registers from different groups cannot be combined for one transfer. To update all Sound Generator registers, six 16-bit DMA transfers are necessary.

25.3 Operation of the Sound Generator

The following section describes the operation of the various blocks in the Sound Generator.

Register reload operation

To support synchronised on-the-fly reloading of the registers, the Sound Generator uses the shadow register concept.

Each Sound Generator reload register has a corresponding shadow register. The CPU/DMA can write to the reload register at any time but copying the reload register contents to the shadow register only happens on a Register Reload event. Refer to Figure 25-17.

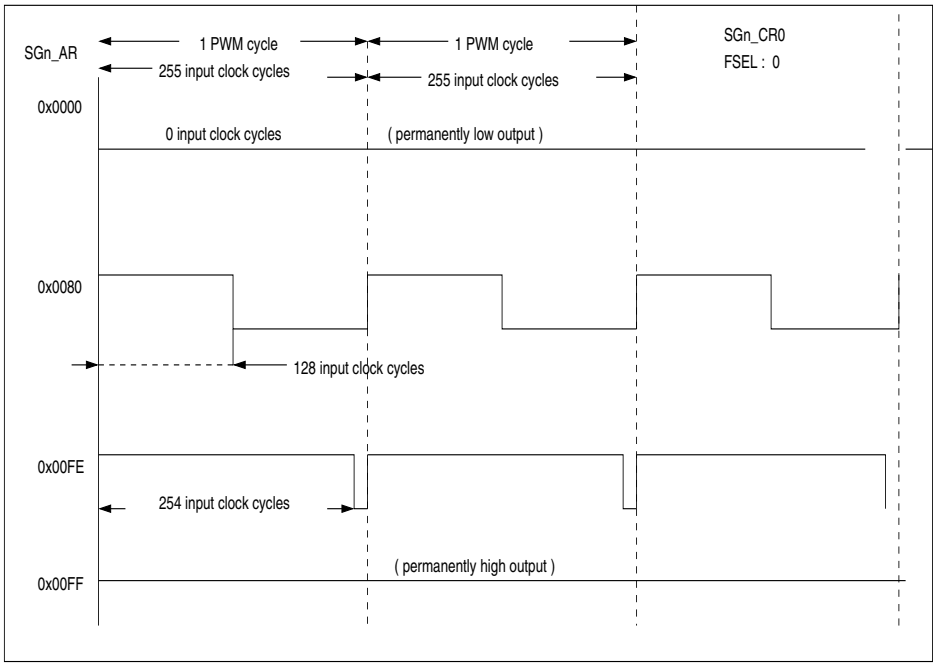
The loading of the shadow registers to the respective counters is synchronised by the tone pulse. Five Sound Generator Reload registers have a corresponding shadow register:

1. SGn_ECRL
2. SGn_FRL
3. SGn_ARL
4. SGn_TARL
5. SGn_TCRLIDRL

PWM generation

- The Sound Generator generates a PWM signal with a variable period and a duty cycle of 0% (permanently low) to 100% (permanently high)
- The setting of the SGn_CR0:FSEL bit determines if the PWM duty cycle is configured to 255 or 511 cycles
- The PWM duty cycle configured in SGn_ARL is incremented/decremented linearly/exponentially depending on the configuration of the SGn_ECRL:AUTO, SGn_ECRL:IDS and SGn_ECRL:ELS bits according to the following equations:
  - Linear: amp = amp ± (amplitude increment/decrement data shadow register value)
  - Exponential: amp = amp ± (amp/32)

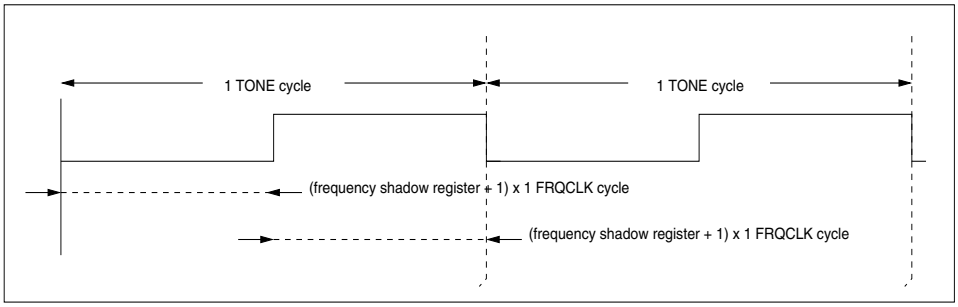
Figure 25-13. PWM pulse cycle timing details



Frequency generation

- The Sound Generator generates a tone or frequency (i.e. SGO) between 100 Hz and 6 kHz with a resolution that is better than 20 Hz at an input frequency of 16 MHz
- It has a 15-bit frequency counter and a toggle flip-flop to generate the output tone for a particular frequency
- The output tone (i.e. the output of the toggle flip-flop) toggles after every (frequency shadow register value + 1) x FRQCLK cycle, as shown in Figure 25-14:

Figure 25-14. Tone pulse timing details



Interrupt, DMA request, and reload generation

- The Sound Generator generates a CPU interrupt, a DMA request and a Register Reload event when:
  - The tone pulse counter (configured by SGn_NRL) reaches zero
  - The amplitude shadow register value matches the target amplitude data register value (configured by SGn_TARL)
  - The amplitude status register (SGn_AR) value becomes zero
- In addition to the above three conditions, a DMA request (optional, controlled by SGn_CR0:DMA) and Register Reload event are also generated during the Sound Generator start operation
- An interrupt, DMA request and Register Reload event generation can be enabled or disabled by programming the respective enable bits in the SGn_ECRL register. Refer to the Sound Generator Extended Control Reload Register (SGn_ECRL) description for interrupt, DMA request and reload enable/disable functionality
- If the target amplitude is set to zero and the amplitude match condition is used for a reload, then the zero amplitude event may not be seen if the amplitude status register (SGn_AR) is reloaded before it goes to zero. Therefore it is recommended to use the same condition for an interrupt or DMA request as for a reload
- Figure 25-15 and Figure 25-16 show the logic for an IRQ interrupt generation and DMA request generation respectively. The logic for a register reload event is shown in Figure 25-17

Figure 25-15. Interrupt request generation logic

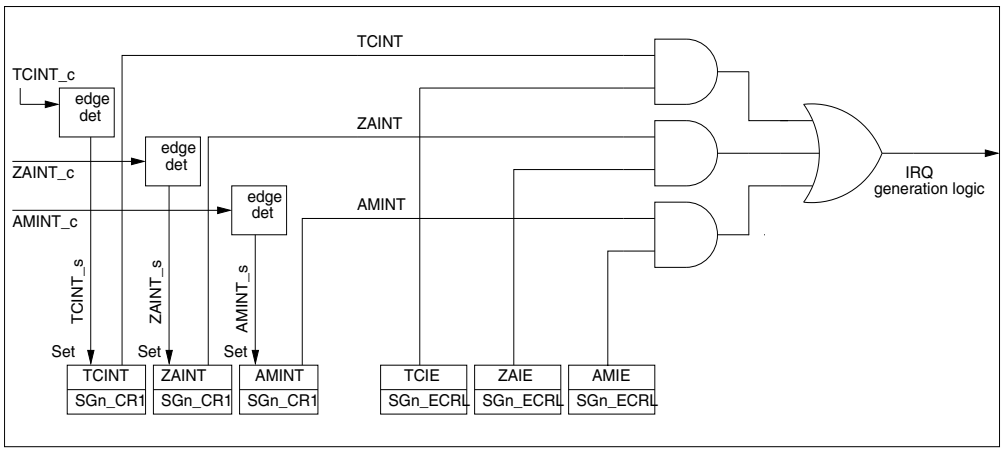


Figure 25-16. DMA request generation logic

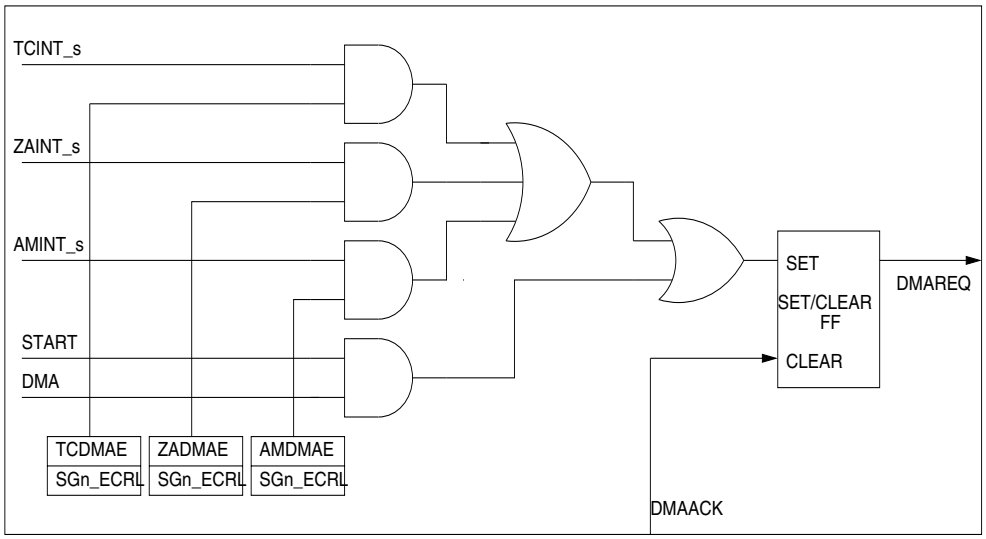
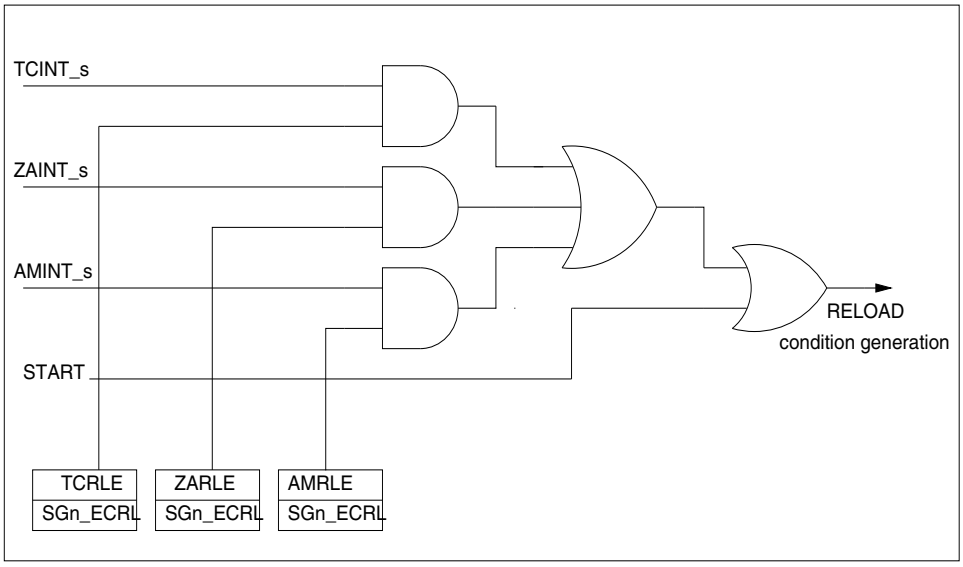


Figure 25-17. Register Reload event generation logic

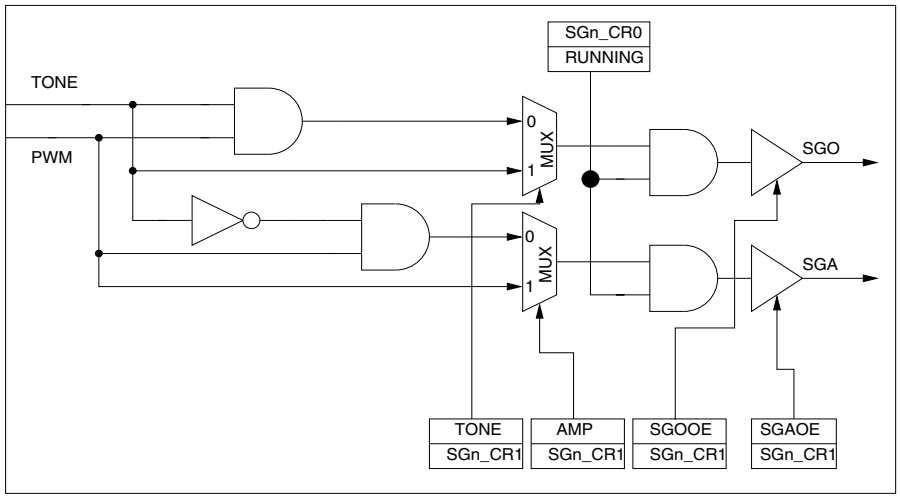


Sound Generator output generation logic

The Sound Generator output generation logic, as shown in Figure 25-18, generates the SGA and SGO outputs. In this figure:

- The TONE and PWM signals can be output separately and mixed outside the device
- It is also possible to output a Mixed-Tone-PWM signal and a Mixed-Inverted-Tone-PWM signal (i.e. PWM mixed with inverted tone)
- The Sound Generator output signals are multiplexed to the SGO and SGA outputs using the SGn_CR1:TONE and SGn_CR1:AMP bits
- All outputs are set to level 'L' when the Sound Generator is in the Stop mode of operation (See 'Sound Generator mode control logic' below)

Figure 25-18. Output generation logic



Sound Generator mode control logic

The Sound Generator has three modes of operation: Stop, Running and Debug mode.

Stop mode—There are two ways to enter Stop mode:

- Setting the SGn_CR0:STOP bit
- When the amplitude register value reaches zero and the automatic stop-on-zero amplitude is enabled by setting SGn_ECRL:SGDADR to ’1’. This clears the SGn_CR0:RUNNING bit to ’0’.

Sound Generator operation is stopped by setting the sound outputs (SGA and SGO) to level ’L’. It can be restarted by reloading the configuration after setting either the SGn_CR0:START bit or the SGn_CR0:RESUME bit in the event that sound generation was paused.

Running mode— In Running mode, the Sound Generator is in the normal mode of operation with sound frequency and amplitude generated on the SGO and SGA outputs.

Debug mode— Debug mode is enabled by setting SGn_CR0:DBGE to '1'. With this setting and the processor in a debug state, the Sound Gemerator can enter the Debug mode of operation. During Debug mode, Sound Generator operation is stopped by masking the SGn_CR0:RUNNING bit, which in turn will stop all the counters and hence sound output. Debug mode is mainly used for software debugging.

For the definition of a debug state, refer to Section 11.8 of the Arm Cortex-R4 Technical Reference Manual.

DMA-based Sound Generator register update operation

This section describes the relationship between the DMA Transfer Update Enable Register (SGn_DER) and the DMA Transfer Indirect Register (SGn_DMAR). In addition, it gives a brief overview of:

- The number of DMA transfers
- DMA transfer size
- Transfer byte positions

The number of DMA transfers

The number of DMA transfers depends on the setting of SGn_DER and is given by:

Number of DMA transfers = (Number of ‘1’s in SGn_DER)/2 (rounded up to the next integer number)

Examples:

1. SGn_DER:CRE0 and SGn_DER:CRE1 bits are set: Number of DMA transfer = 2/2 = 1

2. SGn_DER:CRE0, SGn_DER:ARE0, and SGn_DER:FRE0 bits are set: Number of DMA transfer = 3/2 = 2 (after rounding up)
3. All the bits of SGn_DER are set: Number of DMA transfer = 11/2 = 6 (after rounding up)

DMA transfer size

The size of one DMA transfer is always one half-word (i.e. 2 bytes) irrespective of the SGn_DER settings.

Transfer byte position in the DMA Transfer Indirect Register

The DMA transfer byte position in SGn_DMAR depends on the settings in SGn_DER. The DMA transfer byte position is always right-aligned as shown in Table 25-13, which shows some sample settings of the SGn_DER register.

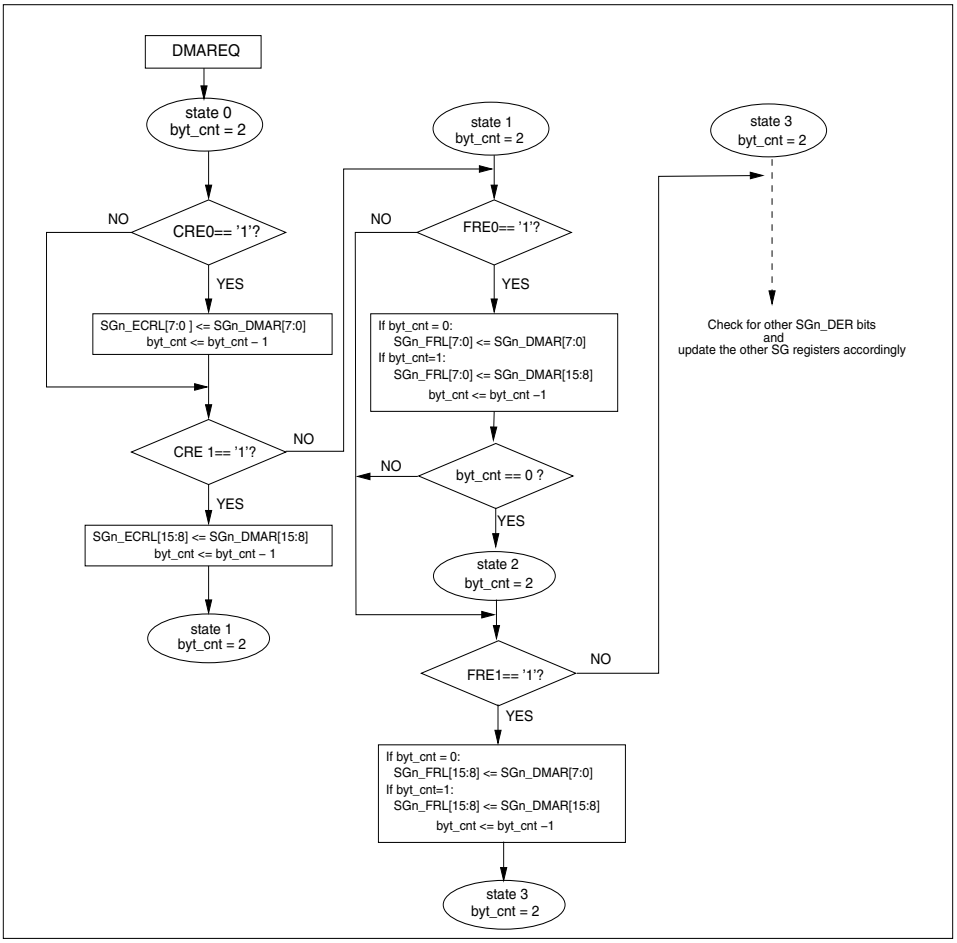
Table 25-13. Sample settings showing the relationship between SGn_DER settings and DMA transfer order

NRE	TCRE	IDRE	TARE1	TARE0	ARE1	ARE0	FRE1	FRE0	CRE1	CRE0	SGn_DMAR [15: 8]	SGn_DMAR [7:0]	No. of Transfers
									1	1	SGn_ECRL [15:8]	SGn_ECRL [7:0]	1
					1	1					SGn_ARL [15:8]	SGn_ARL [7:0]	1
								1		1	SGn_FRL [7:0]	SGn_ECRL [7:0]	1
						1	1		1	1	SGn_ECRL [15:8] 1st-transfer	SGn_ECRL [7:0] 1st-transfer	2
											SGn_ARL [7:0] 2nd-transfer	SGn_FRL [15:8] 2nd-transfer	
			1	1	1	1	1	1	1	1	SGn_ECRL [15:8] 1st-transfer	SGn_ECRL [7:0] 1st-transfer	4
											SGn_FRL [15:8] 2nd-transfer	SGn_FRL [7:0] 2nd-transfer	
											SGn_ARL [15:8] 3rd-transfer	SGn_ARL [7:0] 3rd-transfer	
											SGn_TARL [15:8] 4th-transfer	SGn_TARL [7:0] 4th-transfer	

DMA transfer flowchart

The flowchart in Figure 25-19 shows the updating of the Sound Generator registers using DMA. DMA transfer flow is shown for SGn_ECRL and SGn_FRL register update, and the state machine follows the sequence for updating the rest of the registers based on the respective enable bit settings in the SGn_DER register.

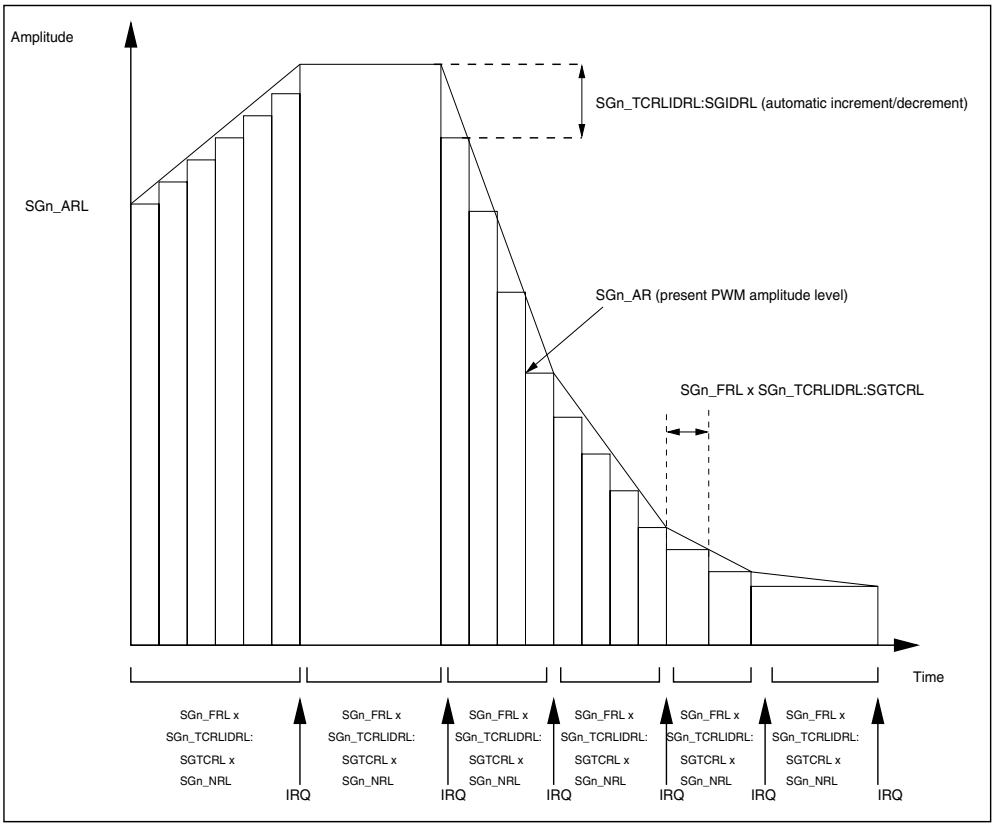
Figure 25-19. DMA-based Sound Generator register's update FSM



Steps in programming the Sound Generator module

The Sound Generator can be programmed by the CPU for sound generation in single or continuous operation mode depending on the sound output requirement. Single and continuous operation can be operated by the CPU or DMA or both, depending on the application. A DMA request can be used to update the Control and Data Registers through a DMA transfer.

Figure 25-20. Conceptual Sound Generation operation



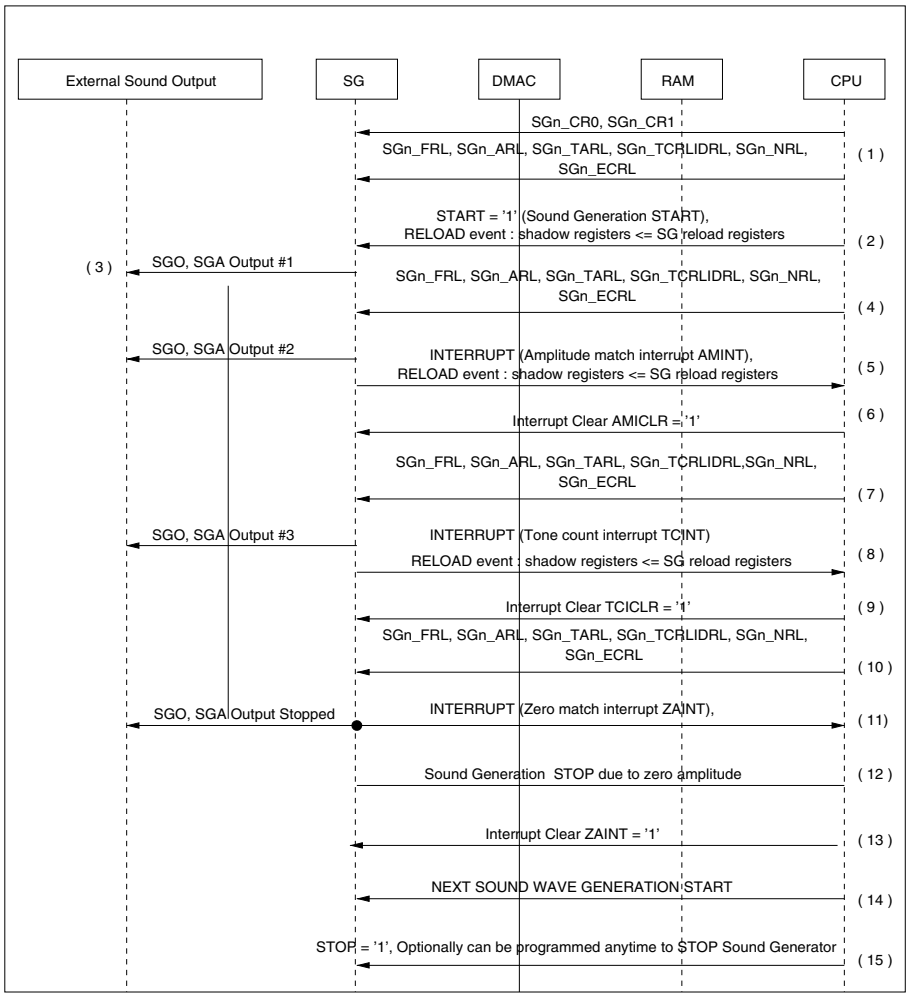
Sound Generator operation steps/flow controlled by CPU

With reference to Figure 25-21, the following steps describe the Sound Generator operation by the CPU:

1. First of all, program the Sound Generator Control Registers (SGn_CR0, SGn_CR1) through software. Then program all the Reload Registers, i.e. the Amplitude Data Reload Register (SGn_ARL), the Frequency Data Reload Register (SGn_-FRL), the Tone Output Number Reload Register (SGn_NRL), the Time Cycle and Increment or Decrement Data Reload Register (SGn_TCRLIDRL) and the Target Amplitude Data Reload Register (SGn_TARL) through software.
2. Set the start bit SGn_CR0:START to '1' which generates a Register Reload event, which in turn updates the shadow registers from the reload registers.
3. The Sound Generator outputs, SGO and SGA, start.
4. The CPU reprograms the necessary Sound Generator registers.
5. Depending on the Amplitude Match Interrupt Enable bit (SGn_ECRL:AMIE), an interrupt is generated when the amplitude register value matches with the target amplitude data shadow register value (configured by SGn_TARL). The next cycle of sound generation starts as the Sound Generator reloads new configuration data from the reload registers to the shadow registers.
6. The CPU then clears the interrupt bit.
7. The CPU reprograms the necessary Sound Generator registers.
8. Depending on the Tone Count Interrupt Enable bit (SGn_ECRL:TCIE), an interrupt is generated when the tone pulse counter (configured by SGn_NRL) reaches zero. The next cycle of sound generation starts as the Sound Generator reloads new configuration data from the reload registers to the shadow registers.
9. The CPU clears the interrupt bit.
10. The CPU programs the necessary Sound Generator registers

11. Depending on the Zero Amplitude Interrupt Enable (SGn_ECRL:ZAIE) bit, an interrupt is generated when the amplitude value reaches zero. The next sound generation cycle starts as the Sound Generator reloads new configuration data from the reload registers to the shadow registers.
12. When the amplitude register reaches zero, sound generation is stopped.
13. The CPU clears the interrupt bit.
14. The Sound Generator is ready to be programmed for new sound wave generation.
15. The SGn_CR0:STOP bit can be set to ‘1' by the CPU anytime during the above steps and this immediately stops sound generation.

Figure 25-21. Control flow diagram describing Sound Generation by the CPU



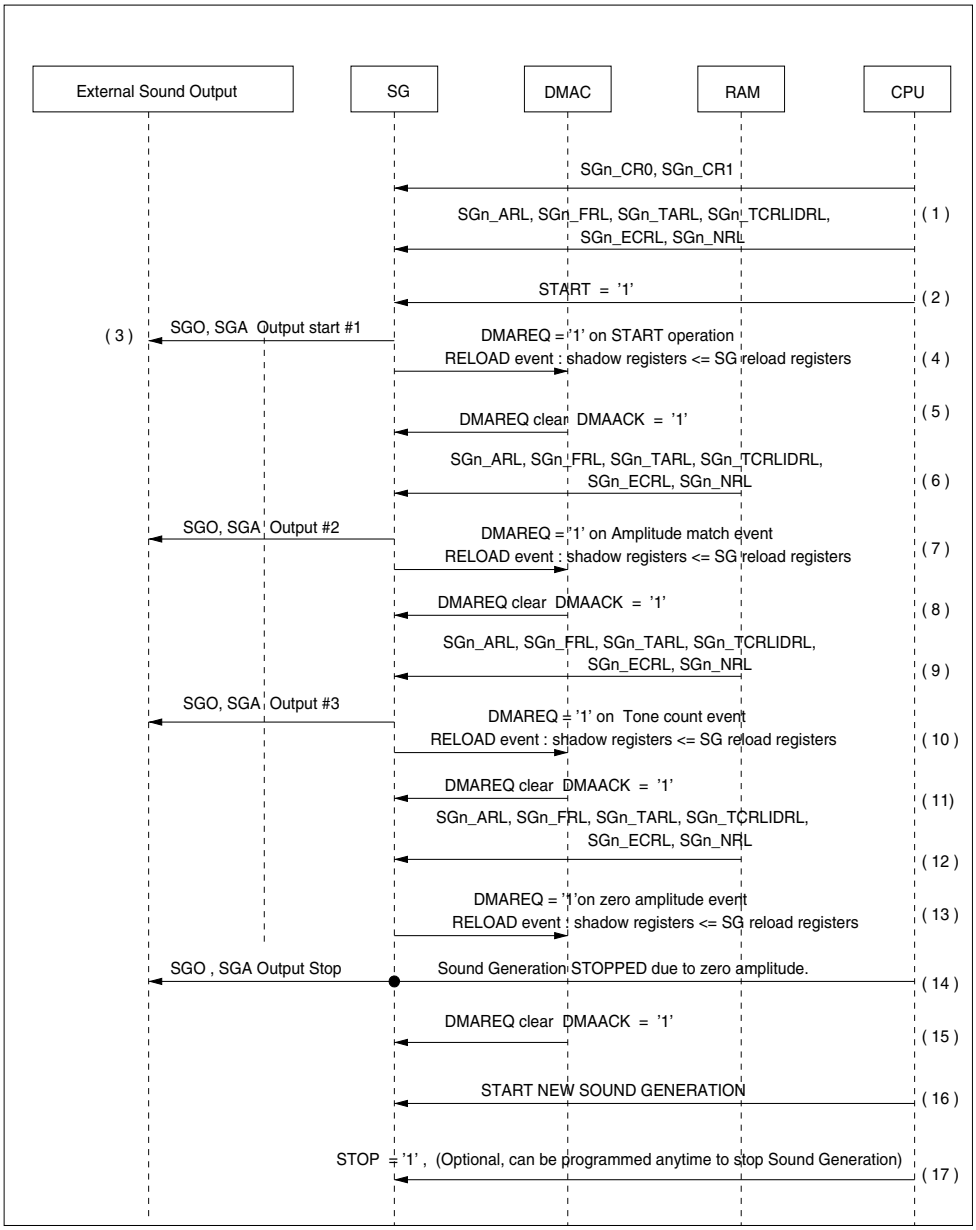
Sound Generator operation steps/flow and register update by DMA

With reference to Figure 25-22, the following steps describe Sound Generator operation and register update by DMA:

1. First of all, program the Sound Generator Control Registers (SGn_CR0, SGn_CR1) through software. The SGn_CR0:DMA bit has to be set to '1' to trigger a DMA request on the START condition. Then program all the Reload Registers, i.e. Amplitude Data Reload Register (SGn_ARL), the Frequency Data Reload Register (SGn_FRL), the Tone Output Number Reload Register (SGn_NRL), the Time Cycle & Increment/Decrement Data Reload Register (SGn_T-CRLIDRL) and the Target Amplitude Data Reload Register (SGn_TARL) through software.
2. Set the start bit SGn_CR0:START to '1' which generates a Register Reload event, which in turn updates the shadow registers from the reload registers. This also triggers a DMA request.

3. The Sound Generator outputs, SGO and SGA, start.
4. DMAREQ is generated because SGn_CR0:START = '1'.
5. DMAREQ is cleared when DMAACK = '1' from the DMA controller.
6. The DMA controller updates the reload registers depending on the settings in the SGn_DER register.
7. Depending on the Amplitude Match DMA Request Enable (SGn_ECRL:AMDMAE) bit, a DMA request is generated when the amplitude register value matches the target amplitude data shadow register value (configured by SGn_TARL). The next sound generation cycle starts as the Sound Generator reloads new configuration data from the reload registers to the shadow registers.
8. DMAREQ is cleared when DMAACK = '1' from the DMA controller.
9. The DMA controller updates the reload registers depending on the settings in SGn_DER.
10. Depending on the Tone Pulse Count DMA Request Enable (SGn_ECRL:TCDMAE) bit, a DMA request is generated when the tone pulse counter (configured by SGn_NRL) reaches zero. The next sound generation cycle starts as the Sound Generator reloads new configuration data from the reload registers to the shadow registers.
11. DMAREQ is cleared when DMAACK = '1' from the DMA controller.
12. The DMA controller updates the reload registers depending on the settings in SGn_DER.
13. Depending on the Zero Amplitude Match DMA Request Enable (SGn_ECRL:ZADMAE) bit, a DMA request is generated when the amplitude value reaches zero. The next sound generation cycle starts as the Sound Generator reloads new configuration data from the reload registers to the shadow registers.
14. When the amplitude register reaches zero, sound generation is stopped.
15. DMAREQ is cleared when DMAACK = '1' from the DMA controller.
16. The Sound Generator is ready to be programmed for new sound wave generation.
17. The SGn_CR0:STOP bit can be set to ‘1' by the CPU anytime during the above steps and this immediately stops sound generation.

Figure 25-22. Sound Generation, DMA register update flow diagram

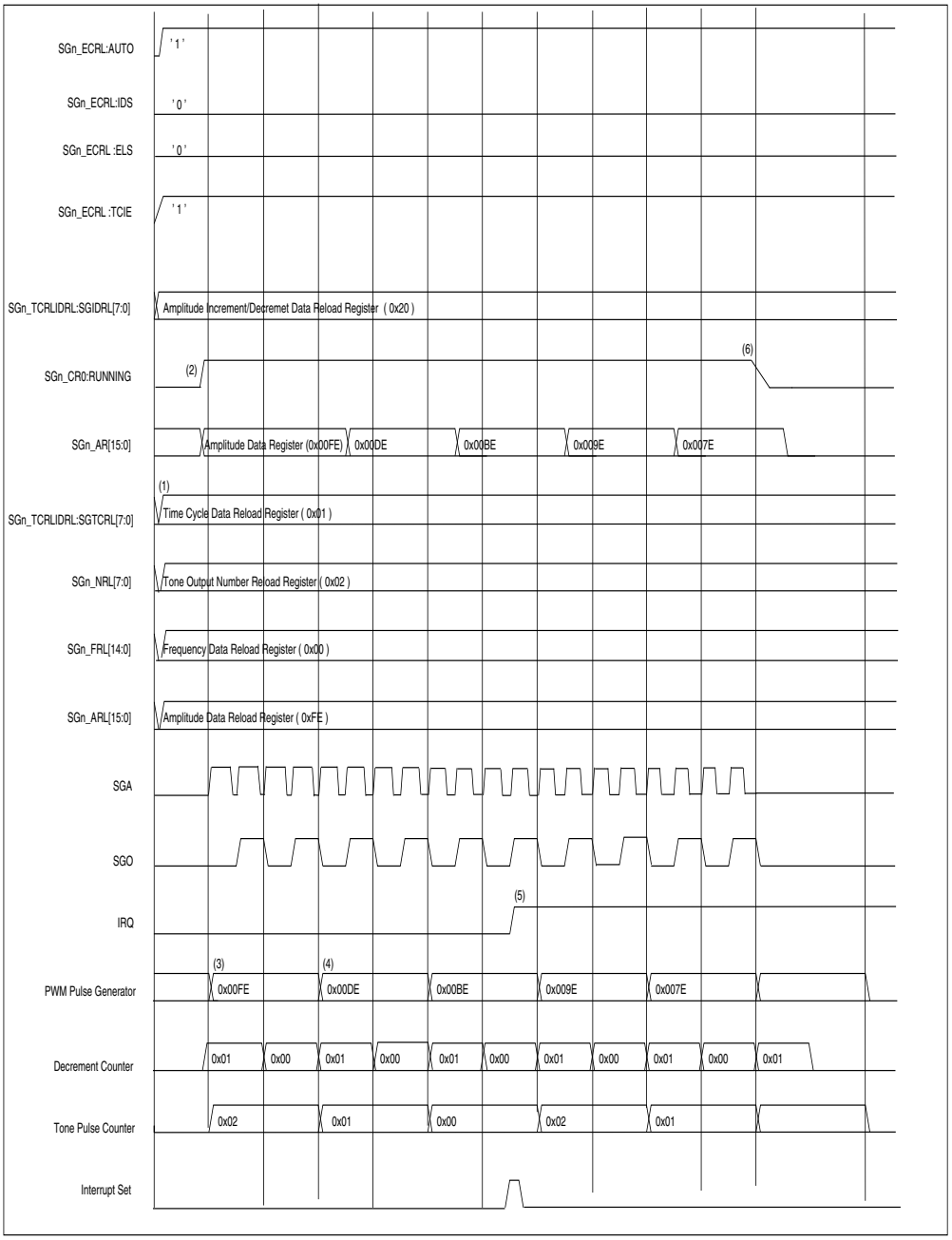


Sound Generator operation (timing)

1. Write the reload values to the Amplitude Data Reload Register (SGn_ARL), the Frequency Data Reload Register (SGn_-FRL), the Tone Output Number Reload Register (SGn_NRL) and the Time Cycle Data Reload Register & Data Increment/ Decrement Data Reload Register (SGn_TCRLIDRL) through software.
2. Initialize the interrupt status bits (SGn_CR1:TCINT, SGn_CR1:AMINT and SGn_CR1:ZAINT) and set the interrupt enable bits (SGn_ECRL:TCIE, SGn_ECRL:AMIE, and SGn_ECRL:ZAIE).
3. Set the start bit SGn_CR0:START to '1' - this also sets the SGn_CR0:RUNNING bit to '1', which indicates that the Sound Generator is running.
4. With SGn_CR0:START set to ‘1’, the SGn_ARL register value is loaded into the PWM pulse generator, the SGn_FRL register value is loaded into the frequency counter, the SGn_NRL value is loaded into the tone pulse counter and the SGn_T-CRLIDRL:SGTCRL register value is loaded into the decrement counter.

5. The decrement counter decrements on the negative edge of the tone pulse. An underflow condition in the decrement counter enables the decrement operation in the tone pulse counter. This condition also enables the amplitude increment/ decrement logic to recalculate the amplitude value for the PWM counter depending on the setting of the automatic increment/ decrement control bits (SGn_ECRL:AUTO/IDS/ELS).
6. When the tone pulse counter counts to the number of tone pulses determined by the values of the SGn_NRL and SGn_TCRLIDRL:SGTCRL registers (when the tone pulse counter is 0x00, the decrement counter is 0x00 and SGO turns from level 'L' to 'H'), an interrupt setting request is generated. The interrupt status bit (SGn_CR1:TCINT) is then set and an interrupt request generated.
7. Set the SGn_CR0:STOP to '1' to stop the Sound Generator from generating sound. The SGn_CR0:RUNNING bit is then cleared to indicate that sound generation has actually stopped.

Figure 25-23. Sound Generator operation timing diagram



26. 16-Bit I/O Timer

This chapter explains the function and operation of the 16-bit I/O Timer.

26.1 Outline of the 16-bit I/O Timer

The 16-bit I/O Timer consists of a 16-bit Free Running Timer (FRT), Output Compare Unit (OCU), and an Input Capture Unit (ICU). It is used for the generation of pulse sequences and to measure the time duration between external events.

Free Running Timer (FRT)

The 16-bit FRT consists of an up/down counter, a control register, a 16-bit compare clear register, a 16-bit compare clear buffer register and a prescaler. The output value obtained from this counter is used as the timer base of Input Capture Unit (ICU) and Output Compare Unit (OCU).

Output Compare Unit (OCU)

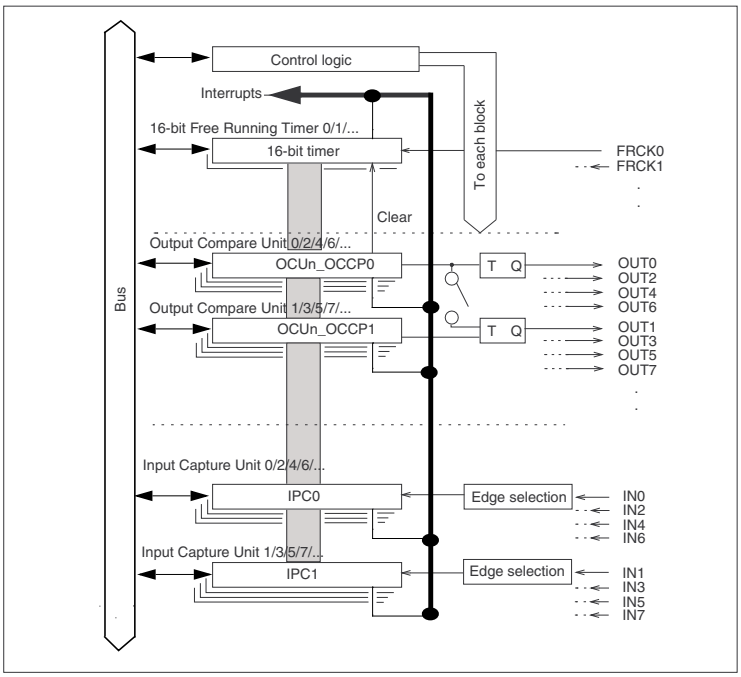
The OCU consists of two 16-bit compare registers, two pairs of compare buffer registers for each output compare register, two compare output pins and control registers. If the value written to the output compare register of this module matches the 16-bit FRT value, the output level of the pin can be toggled and an interrupt can be issued.

Input Capture Unit (ICU)

One ICU consists of two input capture registers and one control register. The ICU detects rising, falling or both edges of an external input signal and stores the 16-bit FRT value at that time in a register. In addition, the ICU can generate an interrupt upon detection of an edge.

Block diagram of the 16-bit I/O Timer

Figure 26-1. Block diagram of the 16-bit I/O Timer



Note: Refer to 26.2 Free Running Timer (FRT) to check the connections between the ICU, OCU and FRT.

26.2 Free Running Timer (FRT)

This section describes the features and the block diagram of the FRT.

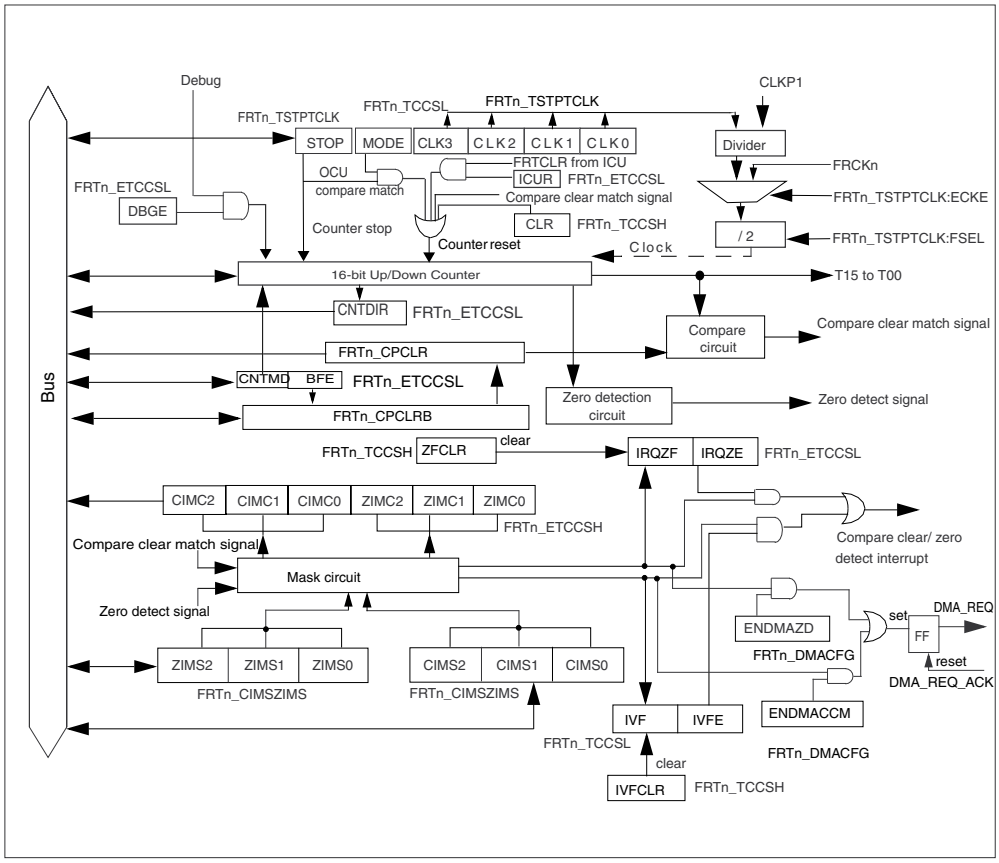
Features of the FRT

The 16-bit FRT consists of an up/down counter, a control register, a 16-bit compare clear register, 16-bit compare clear buffer register,and a prescaler. The output value obtained from this counter is used as the timer base of ICU and OCU. Features of the FRT:

- Eleven counter clocks are available when the internal clock source is selected
- An interrupt can be generated in the event of:
  - Counter overflow
  - Match with compare register 0 and 4 of the OCU
  - Counter reaches zero value. This is called zero-detect interrupt
  - Counter value matches the compare clear register value. This is called a compare clear match interrupt
- The counter value can be initialized to 0x0000 upon a reset, software clear, or if compare clear match is generated between the timer and the compare clear register value only in up-count mode or OCU compare match input or ICU clear input
- It is possible to mask zero-detection, compare clear match interrupt or both so that an interrupt occurs only after several occurrences of the event
- During the up/down count mode, the counter starts the counting down after reaching the compare clear match value
- Programmable timer period up to 1 sec at 64 MHz input clock
- Support for MCU debug mode
- Support for DMA transfer

Block diagram of the 16-bit FRT

Figure 26-2. Block diagram of the 16-bit FRT



26.2.1 16-bit FRT registers

All registers in the 16-bit FRT module are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of the 16-bit FRT

The following registers are available for each instance of the 16-bit FRT:

- Data Register (FRTn_TCDT)
- Compare Clear Buffer Register (FRTn_CPCLRB)
- Compare Clear Register (FRTn_CPCLR)
- Control Status Register (FRTn_TCCS)
- Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK)
- Extended Control Status Register (FRTn_ETCCS)
- Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS)
- DMA Configuration Register (FRTn_DMACFG)

Memory layout of the 16-bit FRT registers

Table 26-1. Memory layout of 16-bit FRT registers

Offset	+1	+0
0x00000000	FRTn_TCDT 00000000 00000000	
0x00000002	FRTn_CPCLRB 11111111 11111111	
0x00000004	FRTn_CPCLR 11111111 11111111	
0x00000006	FRTn_TCCS 00000000 00000000	
0x00000008	FRTn_TSTPTCLK 01000000 00000000	
0x0000000A	FRTn_ETCCS 00000000 00000000	
0x0000000C	FRTn_CIMSZIMS 00000000 00000000	
0x0000000E	reserved XXXXXXXX	FRTn_DMACFG 00000000

26.2.1.1 Data Register (FRTn_TCDT)

This section describes the Data Register (FRTn_TCDT) of the 16-bit FRT.

Data Register (FRTn_TCDT)

Figure 26-3. Data Register (FRTn_TCDT)

FRTn_TCDT															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TCDT[15]	TCDT[14]	TCDT[13]	TCDT[12]	TCDT[11]	TCDT[10]	TCDT[9]	TCDT[8]	TCDT[7]	TCDT[6]	TCDT[5]	TCDT[4]	TCDT[3]	TCDT[2]	TCDT[1]	TCDT[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-2. Data Register (FRTn_TCDT) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	TCDT	Data Register The Data Register can read the counter value of the 16-bit FRT. The counter value is cleared to 0x0000 upon reset. The counter value can be set by writing a value to this register.

Notes:

1. The counter value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (FRTn_TSTPTCLK:STOP = '1'). The Data Register must be accessed by the half-word access instructions.
2. The 16-bit FRT is initialized when the following occurs:
   - Reset
   - Clear bit (FRTn_TCCS:CLR) in Control Status Register
   - ICU clear signal that can be enabled or disabled
   - A match between the Compare Clear Register and the counter value only in up-count mode
   - A match between the Output Compare Register of the OCU and the counter value only in up-count mode

26.2.1.2 Compare Clear Buffer Register (FRTn_CPCLRB)

The Compare Clear Buffer Register (FRTn_CPCLRB) is the 16-bit buffer register for the Compare Clear Register (FRTn_CPCLR).

Compare Clear Buffer Register (FRTn_CPCLRB)

Figure 26-4. Compare Clear Buffer Register (FRTn_CPCLRB)

FRTn_CPCLRB															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CPCLRB[15]	CPCLRB[14]	CPCLRB[13]	CPCLRB[12]	CPCLRB[11]	CPCLRB[10]	CPCLRB[9]	CPCLRB[8]	CPCLRB[7]	CPCLRB[6]	CPCLRB[5]	CPCLRB[4]	CPCLRB[3]	CPCLRB[2]	CPCLRB[1]	CPCLRB[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 26-3. Compare Clear Buffer Register (FRTn_CPCLRB) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	CPCLRB	Compare Clear Buffer Register The Compare Clear Buffer Register is a buffer register for the Compare Clear Register (FRTn_CPCLR). When the buffer function is disabled (FRTn_ETCCSL:BFE = '0') or the FRT is stopped, the value of the Compare Clear Buffer Register is transferred to the Compare Clear Register immediately. If the buffer function is enabled (FRTn_ETCCSL:BFE = '1'), the value is transferred to the Compare Clear Register when the counter value 0 of the 16-bit FRT is detected. To access this register, use a half-word access instruction.

26.2.1.3 Compare Clear Register (FRTn_CPCLR)

The Compare Clear Register (FRTn_CPCLR) is used to compare with the counter value of the 16-bit FRT.

Compare Clear Register (FRTn_CPCLR)

Figure 26-5. Compare Clear Register (FRTn_CPCLR)

FRTn_CPCLR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CPCLR[15]	CPCLR[14]	CPCLR[13]	CPCLR[12]	CPCLR[11]	CPCLR[10]	CPCLR[9]	CPCLR[8]	CPCLR[7]	CPCLR[6]	CPCLR[5]	CPCLR[4]	CPCLR[3]	CPCLR[2]	CPCLR[1]	CPCLR[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 26-4. Compare Clear Register (FRTn_CPCLR) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	CPCLR	Compare Clear Register. The Compare Clear Register is used to compare with the counter value of the 16-bit FRT. In up-count mode, if the 16-bit FRT counter value matches the value in this register, the 16-bit Free Running Timer is reset to 0 at the next count clock edge. If this register is set to 0xFFFF (initial value) in up-count mode, the behavior is equivalent to a counter overflow. In the up/down count mode, the 16-bit FRT changes its mode from the counting up to counting down when this register matches the counter value of the 16-bit FRT. To read this register, use a half-word access instruction.

26.2.1.4 Control Status Register (FRTn_TCCS)

The Control Status Register (FRTn_TCCS) sets the operation mode of the 16-bit FRT and controls interrupts. It is used to clear the timer, compare clear match interrupt and zero-detect interrupts.

Control Status Register (FRTn_TCCS)

Figure 26-6. Control Status Register (FRTn_TCCS)

FRTn_TCCS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	ZFCLR	IVFCLR	CLR	IVF	IVFE	read0	MODE	read0	read0	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp	RpWp	Rp0	RpWp	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-5. Control Status Register (FRTn_TCCS) bits

Bit Position	Bit Field Name	Bit Description
[15:11]	read0	-
[10]	ZFCLR	IRQZF Interrupt Clear This bit is the clear bit for a zero-detect interrupt request flag. '0': No effect '1': Clears the FRTn_ETCCS:IRQZF register bit Reading this bit always returns '0'.
[9]	IVFCLR	IVF Interrupt Clear This bit is the clear bit for the compare clear match interrupt request flag. '0': No effect '1': Clears FRTn_TCCS:IVF register bit Reading this bit always returns '0'.
[8]	CLR	Clear Timer This bit initializes the operating FRT to the value 0x0000. '0': No effect '1': Counter is initialized **Note:** To initialize the counter value while the timer is stopped (FRTn_TSTP:STOP = '1'), write '0000' to the Data Register (FRTn_TCDT).

Table 26-5. Control Status Register (FRTn_TCCS) bits

Bit Position	Bit Field Name	Bit Description
[7]	IVF	Interrupt Request Flag bit for Compare Clear This bit is the interrupt request flag bit for compare clear match or if the counter value matches with the first Output Compare Register of the connected OCU module. This is a read-only bit. This bit is set to '1' if one of the following condition is met: 1. FRT value matches the Compare Clear Register value and compare clear interrupt mask counter equals '000'. 2. FRT value matches the first Output Compare Register of the connected OCU module and the FRTn_TCCS:MODE bit is set to '1'. This bit is cleared by writing '1' to FRTn_TCCS:IVFCLR register. **Note:** If FRTn_CPCLR is set to 0xFFFF and FRTn_CIMSZIMS:CIMS[2:0] is set to '000', this bit behaves as a counter overflow flag.
[6]	IVFE	Interrupt Enable bit for Compare Clear This bit enables the interrupt request for compare clear match or if the counter value matches first Output Compare Register of the connected OCU module. '0': Interrupt disabled '1': Interrupt enabled
[5]	read0	-
[4]	MODE	Set Reset Condition of Timer '0': Initialization of the counter by reset, clear bit or match with the Compare Clear Register(FRTn_CPCLR) in up-count mode '1': Initialization of the counter by reset, clear bit or match with the Compare Clear Register (FRTn_CPCLR) in up-count mode or first Compare Register of the connected OCU module
[3:0]	read0	-

26.2.1.5 Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK)

The Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK) is used to stop the timer. It also sets the clock division ratio and selects the clock source.

Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK)

Figure 26-7. Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK)

FRTn_TSTPTCLK															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ECKE	FSEL	read0	read0	CLK[3]	CLK[2]	CLK[1]	CLK[0]	read0	read0	read0	read0	read0	read0	read0	STOP
RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-6. Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK) bits

Bit Position	Bit Field Name	Bit Description
[15]	ECKE	External Clock Enable This bit is used to select the internal or external clock source. Change this bit status while the Output Compare or Input Capture is stopped because the clock may change after writing data to the ECKE bit. '0': Internal clock source enabled '1': External clock (FRCK pin) enabled **Notes:** 1. If the internal clock is selected, set bits FRTn_TSTPTCLK:CLK[3:0] to the count clock value. This count clock becomes the basic clock. 2. If the clock source is input from the FRCK pin, configure the corresponding port pin correctly. For the minimum pulse width length of FRCK refer to the device-specific datasheet.

Table 26-6. Timer Stop/Timer Clock Configuration Register (FRTn_TSTPTCLK) bits

Bit Position	Bit Field Name	Bit Description
[14]	FSEL	Frequency Selection This bit is used to control division of the counter clock source. Change this bit status while the Output Compare or Input Capture is stopped. The FSEL bit is used to specify the count clock division ratio. '0': The counter clock (as specified by FRTn_TSTPTCLK:ECKE bit) is divided by two '1': The counter clock (as specified by FRTn_TSTPTCLK:ECKE bit) is passed directly
[13:12]	read0	-
[11:8]	CLK	Count Clock Selection These bits are used to select the count clock for the 16-bit FRT. The clock is updated immediately after a value is written to these bits. Therefore, ensure that the Input Capture and Output Compare operations are stopped before a value is written to these bits (for more details refer to Table 26-10)
[7:1]	read0	-
[0]	STOP	Stop Timer This bit is used to stop the timer. '0': Counter enabled (operation) '1': Counter disabled (stop)

26.2.1.6 Extended Control Status Register (FRTn_ETCCS)

The Extended Control Status Register (FRTn_ETCCS) sets the buffer operation mode of the compare clear register, the count mode of the counter, enables the ICU clear function, enables the zero-detect interrupt and enables debug mode. It also stores the values of the interrupt mask counters.

Extended Control Status Register (FRTn_ETCCS)

Figure 26-8. Extended Control Status Register (FRTn_ETCCS)

FRTn_ETCCS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	CIMC[2]	CIMC[1]	CIMC[0]	read0	ZIMC[2]	ZIMC[1]	ZIMC[0]	read0	DBGE	CNTDIR	ICUR	BFE	IRQZF	IRQZE	CNTMD
Rp0	Rp	Rp	Rp	Rp0	Rp	Rp	Rp	Rp0	RpWp	Rp	RpWp	RpWp	Rp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-7. Extended Control Status Register (FRTn_ETCCS) bits

Bit Position	Bit Field Name	Bit Description
[15]	read0	-
[14:12]	CIMC	Compare Clear Mask Counter Read bits These bits are read-only bits. Reading these bits returns the actual value of the mask counter for compare clear match.
[11]	read0	-
[10:8]	ZIMC	Zero Detect Mask Counter Read bits These bits are read-only bits. Reading these bits returns the actual value of the mask counter for zero detect.
[7]	read0	-
[6]	DBGE	Debug Enable This bit is for debug mode support. '0': Debug mode Disable '1': Debug mode Enable When DBGE is Set to 1 and the processor is in debug state, timer operation is stopped. For definition of debug state, refer to section 11.8 of the Arm® Cortex®-R4 technical reference manual.

Table 26-7. Extended Control Status Register (FRTn_ETCCS) bits

Bit Position	Bit Field Name	Bit Description
[5]	CNTDIR	Count Direction This is a read-only bit and it gives the information regarding the count direction of FRT. '0': Counter is currently counting in up direction '1': Counter is currently counting in down direction Writing to this bit is ignored.
[4]	ICUR	FRT Counter Reset by ICU Enable This bit enables the counter reset by ICU. '0': FRT counter reset by ICU is disabled '1': FRT counter reset by ICU is enabled This function is independent of the FRTn_TCCS:MODE bit setting. **Note:** Enable corresponding register control bit of ICU to use this function.
[3]	BFE	Compare Clear Buffer Enable '0': Compare Clear Buffer function is disabled '1': Compare Clear Buffer function is enabled
[2]	IRQZF	Interrupt Request Flag for Zero Detect This bit is the zero-detect interrupt request flag bit. This is a read-only bit. This bit is set to '1' if the FRT value reaches 0x0000 in up/down-count mode and the zero-detect interrupt mask counter equals '000'. This bit is cleared by writing '1' to FRTn_TCCS:ZFCLR. **Note:** This bit is not set to '1' if the 16-bit FRT is cleared by writing '1' to FRTn_TCCS:CLR.
[1]	IRQZE	Zero Detect Interrupt Enable This bit enables the zero-detect interrupt request. '0': Interrupt disabled, '1': Interrupt enabled
[0]	CNTMD	Count Mode of FRT The bit is used to select a count mode of the Free Running Timer. '0': Selects the up-count mode '1': Selects the up/down count mode This bit can be written if the timer is operating or stopped. When the timer is running, the value written to this bit is stored in a buffer and the count mode changes based on the buffer value the next time the counter value goes to 0x0000.

26.2.1.7 Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS)

The Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS) enables the masking function to generate an interrupt only after a number of occurrences of compare clear match. It also enables the masking function to generate an interrupt only after a number of occurrences of zero detect.

Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS)

FRTn_CIMSZIMS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	ZIMS[2]	ZIMS[1]	ZIMS[0]	read0	read0	read0	read0	read0	CIMS[2]	CIMS[1]	CIMS[0]
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-8. Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS) bits

Bit Position	Bit Field Name	Bit Description
[15:11]	read0	-
[10:8]	ZIMS	Zero-detect Interrupt Mask Select These bits enable the mask function to generate an interrupt after a number of occurrences of zero detect. When '000' is set to these bits, the zero-detect interrupt is not masked. Reading these bits returns the zero-detect interrupt mask reload value. **Notes:** 1. When FRT is running (FRTn_TSTPTCLK:STOP = '0'), writing value to the mask register is reloaded to the zero-detect interrupt mask counter after the counter reaches '000'. 2. When FRT is stopped (FRTn_TSTPTCLK:STOP = '1'), writing value to the mask register is reloaded to the zero-detect interrupt mask counter immediately.
[7:3]	read0	-

Table 26-8. Compare/Zero-Interrupt Mask Register (FRTn_CIMSZIMS) bits

Bit Position	Bit Field Name	Bit Description
[2:0]	CIMS	Compare Clear Interrupt Mask Select These bits enable the mask function to generate an interrupt after a number of occurrences of compare clear match. When '000' is set to these bits, the compare clear match interrupt is not masked. Reading these bits returns the compare clear match interrupt mask reload value. **Notes:** 1. When FRT is running (FRTn_TSTPTCLK:STOP = '0'), writing value to the mask register is reloaded to the compare clear match interrupt mask counter after the counter reaches '000'. 2. When FRT is stopped (FRTn_TSTPTCLK:STOP = '1'), writing value to the mask register is reloaded to the compare clear match interrupt mask counter immediately.

26.2.1.8 DMA Configuration Register (FRTn_DMACFG)

The DMA Configuration Register (FRTn_DMACFG) contains the bits to enable/disable the DMA request for zero detect and compare clear interrupts.

DMA Configuration Register (FRTn_DMACFG)

Figure 26-9. DMA Configuration Register (FRTn_DMACFG)

FRTn_DMACFG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	ENDMACCM	ENDMAZD
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 26-9. DMA Configuration Register (FRTn_DMACFG) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	ENDMACCM	Compare Clear DMA Enable This bit is used to enable/disable the DMA request for compare clear match interrupt. '0': Compare clear match interrupt DMA request is disabled '1': Compare clear match interrupt DMA request is enabled
[0]	ENDMAZD	Zero Detect DMA Enable This bit is used to enable/disable the DMA request for zero-detect interrupt. '0': Zero-detect interrupt DMA request is disabled '1': Zero-detect interrupt DMA request is enabled

26.2.2 Operation of 16-bit FRT

The 16-bit FRT starts counting from counter value 0x0000 after the reset is released. The counter value is used as the time base for the 16-bit Output Compare and 16-bit Input Capture.

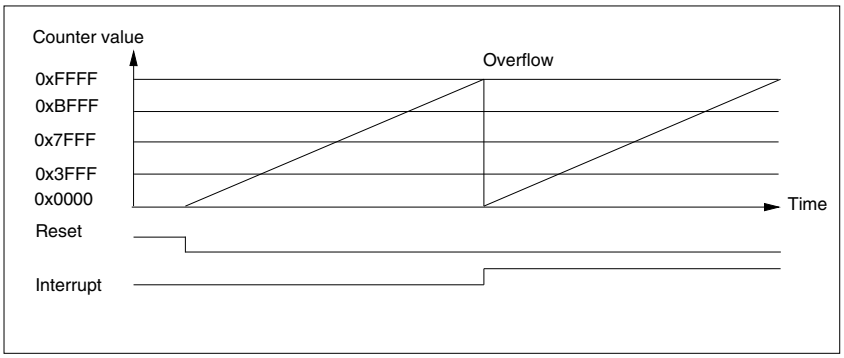
16-bit FRT operation

The counter value is cleared under the following conditions:

- Reset
- When an overflow occurs
- The Compare Clear Register value matches the 16-bit FRT value in up-count mode (the mode must be set)
- When a match with the Output Compare Register occurs (this depends on the mode)
- When '1' is written to the CLR bit of the FRTn_TCCS register during operation
- When 0x0000 is written to the FRTn_TCDT register during stop

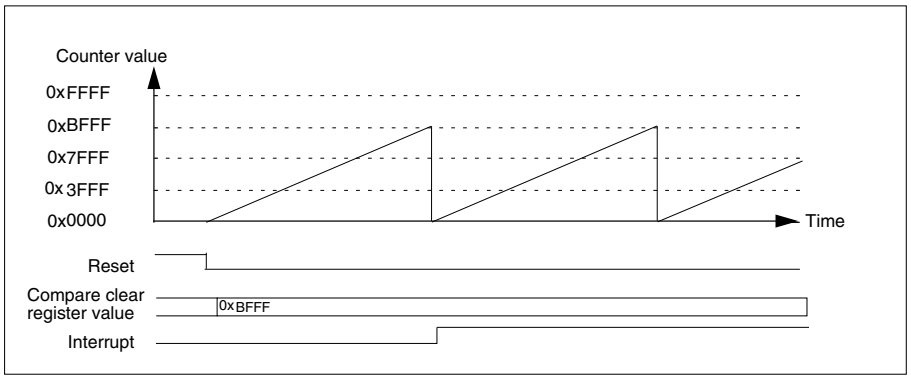
Clearing the counter by an overflow

Figure 26-10. Clearing the counter by an overflow in up-count mode



Clearing the counter at Compare Clear Register value match in up-count mode

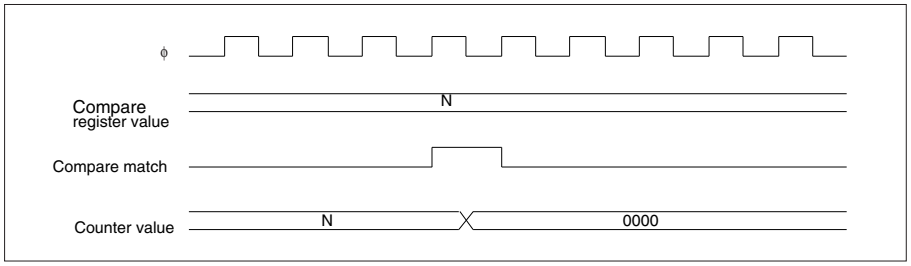
Figure 26-11. Clearing the counter at Compare Clear Register value match in up-count mode



16-bit Free Running Timer clear timing

The counter can be cleared upon a reset, software clear, Compare Clear Register match, match with the Output Compare Register or ICU clear. By a reset, the counter is immediately cleared. By a match with the output compare register or compare clear register or by software clear (FRTn_TCCS:CLR = '1'), the counter is cleared in synchronization with the count timing.

Figure 26-12. 16-bit Free Running Timer clear timing



Timer mode

The following modes can be selected for the 16-bit FRT.

Up-count mode (FRTn_ETCCS:CNTMD = '0')

In the up-count mode, the counter starts counting from the Timer Data Register (FRTn_TCDT) and continues counting up until the count matches the value in the Compare Clear Register (FRTn_CPCLR) or overflow. Then, the counter is cleared to 0x0000 and the counter restarts counting up.

Up/down count mode (FRTn_ETCCS:CNTMD = ’1’)

In the up/down count mode, the counter starts counting from the preset Timer Data Register (FRTn_TCDT) and continues counting up until the count matches the value in the Compare Clear Register (FRTn_CPCLR). Then, counting changes from counting up to counting down, the counter value counts down until it reaches to 0x0000, and the counter restarts counting up.

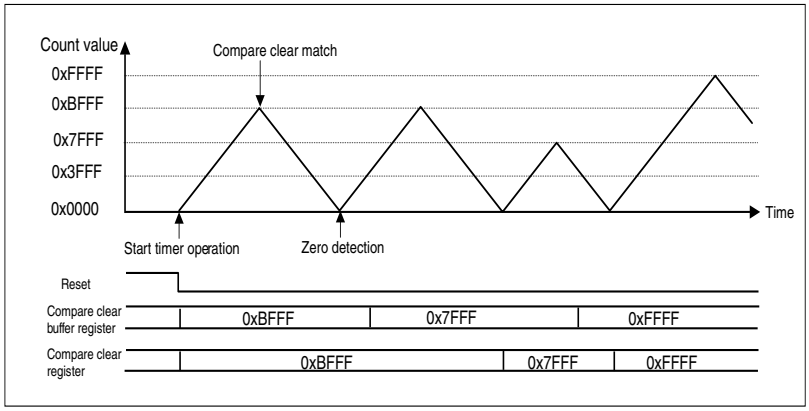
The count mode bit (FRTn_ETCCS:CNTMD) can be set at any time, regardless of whether the timer is running or halted. The value written to the bit when the timer is running is stored in a buffer and the actual count mode does not change until the counter value reaches 0x0000.

Compare clear buffer

The Compare Clear Register (FRTn_CPCLR) has a buffer feature that can be enabled or disabled. When the buffer function is enabled (FRTn_ETCCS:BFE = '1'), data written to the Compare Clear Buffer Register (FRTn_CPCLRB) is transferred to the FRTn_CPCLR register when zero is detected on the 16-bit FRT. When the buffer function is disabled, then data is written directly to the FRTn_CPCLR register.

Operation in up/down count mode when compare clear buffer is enabled

Figure 26-13. Operation in up/down count mode



Timer interrupt

The 16-bit FRT can generate the following interrupts:

1. Compare clear interrupt/overflow interrupt.
2. Zero-detect interrupt.
3. Match with Output Compare Register 0 and 4.

Compare-clear interrupts are generated when the counter value matches the value of the Compare Clear Register.

Zero-detect interrupts are generated when the counter value reaches 0x0000.

Overflow interrupts are generated when the counter value changes from 0xFFFF to 0x0000 in up-count mode.

Overflow interrupt and compare clear interrupt uses the same interrupt enable and same interrupt flag.

Interrupt mask function

It is possible to mask zero detection and compare clear match interrupts.

The zero-detect interrupts are masked by the FRTn_CIMSZIMS:ZIMS[2:0] bits. When the ZIMS[2:0] bits are '000' and zero-detect interrupt is enabled, then periodic interrupt requests are not masked.

The compare-clear interrupts are masked by the FRTn_CIMSZIMS:CIMS[2:0] bits. When the CIMS[2:0] bits are '000' and the compare-clear match interrupt is enabled, then periodic interrupt requests are not masked.

Prescaler unit

The prescaler provides the timer clock by dividing the peripheral clock to a lower frequency.

The prescaler can be programmed with divider values ranging from a minimum of 1 to a maximum of 1024 using the FRTn_TSTPTCLK:CLK[3] to FRTn_TSTPTCLK:CLK[0] bits as given in the following table. The clock is updated immediately after a value is written to these bits.

Table 26-10. Count clock selection table

CLK3	CLK2	CLK1	CLK0	Count clock selection
'0'	'0'	'0'	'0'	ϕ
'0'	'0'	'0'	'1'	ϕ/2
'0'	'0'	'1'	'0'	ϕ/4
'0'	'0'	'1'	'1'	ϕ/8
'0'	'1'	'0'	'0'	ϕ/16
'0'	'1'	'0'	'1'	ϕ/32
'0'	'1'	'1'	'0'	ϕ/64
'0'	'1'	'1'	'1'	ϕ/128
'1'	'0'	'0'	'0'	ϕ/256
'1'	'0'	'0'	'1'	ϕ/512
'1'	'0'	'1'	'0'	ϕ/1024
'1'	'0'	'1'	'1'	reserved
'1'	'1'	'0'	'0'	reserved
'1'	'1'	'0'	'1'	reserved
'1'	'1'	'1'	'0'	reserved

Table 26-10. Count clock selection table

CLK3	CLK2	CLK1	CLK0	Count clock selection
'1'	'1'	'1'	'1'	reserved
Note: ϕ = Peripheral clock				

26.3 Output Compare Unit (OCU)

This section describes the features and the block diagram of the OCU.

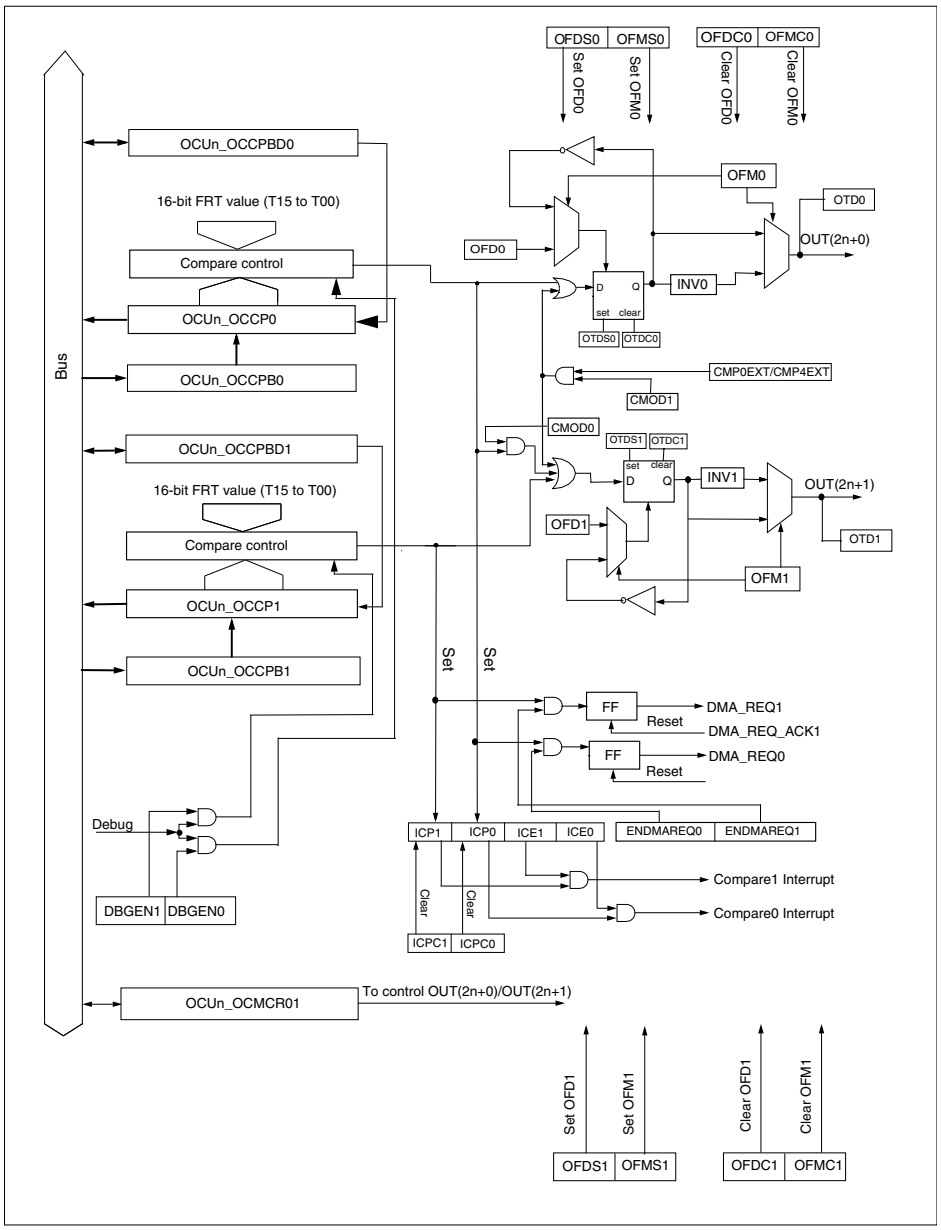
Features of the OCU

The OCU consists of two 16-bit output compare registers, two pairs of output compare buffer registers for each output compare register, two compare output pins and control registers. If the value written to the output compare register of this module matches the 16-bit FRT value, the output level of the pin can be toggled and an interrupt can be issued.

- For each OCU, two output compare registers exist which can be used independently. Depending on the mode setting, the two output compare registers can be used to control pin outputs
- The initial value for each pin output can be specified separately
- An interrupt can be issued upon a match as a result of comparison
- Each OCU can generate one Pulse Width Modulation (PWM) signal using two output compare registers
- The output compare registers can be used to generate one cyclic waveform signal
- Full range features: The output of the OCU does not toggle when the OCU output compare register is 0x0000 or 0xFFFF, greatr than or equal to the FRT compare clear register in full duty cycle mode (OCUn_EOCS01:FDEN = '1')
- When the OCU functions in fixed mode, fixed data is output at compare events
- Two Output Compare Buffer Registers (OCUn_OCCPB, OCUn_OCCPBD) for each OCU channel are used as a buffer register for the Output Compare Register (OCUn_OCCP0/ OCUn_OCCP1)
- One Output Compare Mode Control Register (OCUn_OCMCR01) to control the output level and to invert the output level is available
- The output compare buffer registers of the OCU are updated either by the CPU or DMA
- Support for MCU debug mode

Block diagram of Output Compare Unit

Figure 26-14. Block diagram of Output Compare Unit



Note:

FRT clear by OCU compare match is only supported for:

- OCU0 (default FRT: FRT0)
- OCU4 (default FRT: FRT1)
- OCU8 (default FRT: FRT2)
- OCU10 (default FRT: FRT3)

The above OCUs can reset any selected FRT.

(For availability of the modules refer to the device-specific datasheet.)

26.3.1 16-bit OCU registers

All registers in the 16-bit OCU module are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of 16-bit Output Compare Unit

The following registers are available for each instance of 16-bit Output Compare Unit:

- Output Compare Register (OCUn_OCCP0/OCUn_OCCP1)
- Output Compare Buffer Registers (OCUn_OCCPB0/OCUn_OCCPB1)
- Output Compare Down Buffer Registers (OCUn_OCCPBD0/OCUn_OCCPBD1)
- Control Register (OCUn_OCS01)
- Control Clear Register (OCUn_OCSC01)
- Control Set Register (OCUn_OCSS01)
- Status Register (OCUn_OSR01)
- Status Clear Register (OCUn_OSCR01)
- Extended Output Compare Control Status Register(OCUn_EOCS01)
- Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01)
- Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01)
- Debug Configuration Register (OCUn_DEBUG01)
- DMA Configuration Register (OCUn_DMACFG01)
- Compare Mode Control Register (OCUn_OCMCR01)

Memory layout of 16-bit Output Compare Unit registers

Table 26-11. Memory layout of 16-bit Output Compare Unit registers

Offset	+1	+0
0x00000000	OCUn_OCCP0 00000000 00000000	
0x00000002	OCUn_OCCP1 00000000 00000000	
0x00000004	OCUn_OCCPB0 00000000 00000000	
0x00000006	OCUn_OCCPB1 00000000 00000000	
0x00000008	OCUn_OCCPBD0 00000000 00000000	
0x0000000A	OCUn_OCCPBD1 00000000 00000000	
0x0000000C	OCUn_OCS01 00000000 00000000	
0x0000000E	OCUn_OCSC01 00000000 00000000	
0x00000010	OCUn_OCSS01 00000000 00000000	

Table 26-11. Memory layout of 16-bit Output Compare Unit registers

Offset	+1	+0
0x00000012	reserved XXXXXXXX	OCUn_OSR01 00000000
0x00000014	reserved XXXXXXXX	OCUn_OSCR01 00000000
0x00000016	OCUn_EOCS01 00000000 00000000	
0x00000018	OCUn_EOCSSH01 00000000	reserved XXXXXXXX
0x0000001A	OCUn_EOCSCH01 00000000	reserved XXXXXXXX
0x0000001C	OCUn_DEBUG01 00000000	OCUn_DMACFG01 00000000
0x0000001E	OCUn_OCMCR01 00000000	reserved XXXXXXXX

Note: The suffix 'n' denotes the number of the OCU's (0, 1, 2,...).

26.3.1.1 Output Compare Register (OCUn_OCCP0/OCUn_OCCP1)

The 16-bit Output Compare Registers are compared with the 16-bit FRT. These registers must be accessed by half-word access instructions. When the value of the register matches that of the 16-bit FRT, a compare signal is generated and the output compare interrupt flag is set. If the output is enabled, the output level corresponding to the Output Compare Register is set depending on the configuration.

Output Compare Register (OCUn_OCCP0)

Figure 26-15. Output Compare Register (OCUn_OCCP0)

OCUn_OCCP0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
OCCP[15]	OCCP[14]	OCCP[13]	OCCP[12]	OCCP[11]	OCCP[10]	OCCP[9]	OCCP[8]	OCCP[7]	OCCP[6]	OCCP[5]	OCCP[4]	OCCP[3]	OCCP[2]	OCCP[1]	OCCP[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-12. Output Compare Register (OCUn_OCCP0) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	OCCP	Output Compare Register This register can only be read using half-word accesses. The Output Compare Register is compared with the count value of the 16-bit Free Running Timer. Reading returns the current compare value.

Output Compare Register (OCUn_OCCP1)

Figure 26-16. Output Compare Register (OCUn_OCCP1)

OCUn_OCCP1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
OCCP[15]	OCCP[14]	OCCP[13]	OCCP[12]	OCCP[11]	OCCP[10]	OCCP[9]	OCCP[8]	OCCP[7]	OCCP[6]	OCCP[5]	OCCP[4]	OCCP[3]	OCCP[2]	OCCP[1]	OCCP[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-13. Output Compare Register (OCUn_OCCP1) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	OCCP	Output Compare Register This register can only be read using half-word accesses. The Output Compare Register is compared with the count value of the 16-bit FRT. Reading returns the current compare value.

26.3.1.2 Output Compare Buffer Registers (OCUn_OCCPB0/ OCUn_OCCPB1)

The Output Compare Buffer Registers (OCUn_OCCPB0/OCUn_OCCPB1) are 16-bit buffer registers for the Output Compare Registers (OCUn_OCCP0/OCUn_OCCP1).

Output Compare Buffer Register (OCUn_OCCPB0)

Figure 26-17. Output Compare Buffer Register (OCUn_OCCPB0)

OCUn_OCCPB0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
OCCPB[15]	OCCPB[14]	OCCPB[13]	OCCPB[12]	OCCPB[11]	OCCPB[10]	OCCPB[9]	OCCPB[8]	OCCPB[7]	OCCPB[6]	OCCPB[5]	OCCPB[4]	OCCPB[3]	OCCPB[2]	OCCPB[1]	OCCPB[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-14. Output Compare Buffer Register (OCUn_OCCPB0) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	OCCPB	Output Compare Buffer Register This register can only be accessed using half-word instructions. When the buffer function is disabled (OCUn_EOCS01:BUF0 = '0') the value of the Output Compare Buffer Register is transferred to the Output Compare Register immediately. When the buffer function is enabled (OCUn_EOCS01:BUF0 = '1') then the value is transferred when the compare clear match or zero-detection occurs according to the transfer selection bits (OCUn_EOCS01:BTS[1:0]) after one OCU clock cycle.

Output Compare Buffer Register (OCUn_OCCPB1)

Figure 26-18. Output Compare Buffer Register (OCUn_OCCPB1)

OCUn_OCCPB1															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OCCPB[15]	OCCPB[14]	OCCPB[13]	OCCPB[12]	OCCPB[11]	OCCPB[10]	OCCPB[9]	OCCPB[8]	OCCPB[7]	OCCPB[6]	OCCPB[5]	OCCPB[4]	OCCPB[3]	OCCPB[2]	OCCPB[1]	OCCPB[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-15. Output Compare Buffer Register (OCUn_OCCPB1) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	OCCPB	Output Compare Buffer Register This register can only be accessed using half-word instructions. When the buffer function is disabled (OCUn_EOCS01:BUF1 = '0') the value of the Output Compare Buffer Register is transferred to the Output Compare Register immediately. When the buffer function is enabled (OCUn_EOCS01:BUF1 = '1') then the value is transferred when the compare clear match or zero-detection occurs according to the transfer selection bits (OCUn_EOCS01:BTS[3:2]) after one OCU clock cycle.

Note: Ensure to evaluate the counter value of the FRT or to initialize the timer before writing to this register. If the buffer function is disabled and a compare match coincides with write operation to this register, the resulting compare operation is not predictable.

26.3.1.3 Output Compare Down Buffer Registers (OCUn_OCCPBD0/OCUn_OCCPBD1)

The Output Compare Down Buffer Registers (OCUn_OCCPBD0/OCUn_OCCPBD1) are the 16-bit down buffer registers for the Output Compare Registers OCUn_OCCP0/ OCUn_OCCP1. These registers are used only when the FRT is running in up-down count mode.

Output Compare Down Buffer Register (OCUn_OCCPBD0)

Figure 26-19. Output Compare Down Buffer Register (OCUn_OCCPBD0)

OCUn_OCCPBD0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
OCCPBD[15]	OCCPBD[14]	OCCPBD[13]	OCCPBD[12]	OCCPBD[11]	OCCPBD[10]	OCCPBD[9]	OCCPBD[8]	OCCPBD[7]	OCCPBD[6]	OCCPBD[5]	OCCPBD[4]	OCCPBD[3]	OCCPBD[2]	OCCPBD[1]	OCCPBD[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-16. Output Compare Down Buffer Register (OCUn_OCCPBD0) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	OCCPBD	Output Compare Down Buffer Register This register can only be accessed as half word. This buffer register is used to update the Output Compare Register at compare clear match if the (OCUn_EOCS01:BTS[1:0]) bits are configured correspondingly.

Output Compare Down Buffer Register (OCUn_OCCPBD1)

Figure 26-20. Output Compare Down Buffer Register (OCUn_OCCPBD1)

OCUn_OCCPBD1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
OCCPBD[15]	OCCPBD[14]	OCCPBD[13]	OCCPBD[12]	OCCPBD[11]	OCCPBD[10]	OCCPBD[9]	OCCPBD[8]	OCCPBD[7]	OCCPBD[6]	OCCPBD[5]	OCCPBD[4]	OCCPBD[3]	OCCPBD[2]	OCCPBD[1]	OCCPBD[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-17. Output Compare Down Buffer Register (OCUn_OCCPBD1) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	OCCPBD	Output Compare Down Buffer Register This register can only be accessed using half-word instructions. This buffer register is used to update the compare register at compare clear match if the (OCUn_EOCS01:BTS[3:0]) bits are configured correspondingly.

26.3.1.4 Control Register (OCUn_OCS01)

The Control Register (OCUn_OCS01) sets the operation mode of output compare, starts and stops output compare and sets the external output pins.

Control Register (OCUn_OCS01)

Figure 26-21. Control Register (OCUn_OCS01)

OCUn_OCS01															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CMOD1	read0	read0	CMOD0	read0	read0	OTD1	OTD0	read0	read0	ICE1	ICE0	read0	read0	CST1	CST0
RpWp	Rp0	Rp0	RpWp	Rp0	Rp0	Rp	Rp	Rp0	Rp0	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-18. Control Register (OCUn_OCS01) bits

Bit Position	Bit Field Name	Bit Description
[15]	CMOD1	Define Comparison Mode of the Output Pin This bit defines the operation mode for the pin output signals. Depending on the defined mode the level is toggled upon a match with different compare registers. See Table 26-33 and 26.3.2 Operation of 16-bit Output Compare Unit.
[14:13]	read0	-
[12]	CMOD0	Define Comparison Mode of the Output Pin This bit defines the operation mode for the pin output signals. Depending on the defined mode the level is toggled upon a match with different compare registers. See Table 26-33 and 26.3.2 Operation of 16-bit Output Compare Unit.
[11:10]	read0	-

Table 26-18. Control Register (OCUn_OCS01) bits

Bit Position	Bit Field Name	Bit Description
[9]	OTD1	Output Pin Level Select for Channel 1 The initial value of the OCU output pin is '0'. This bit can be changed only when the compare operation is disabled by configuring the OCUn_OCS01:CST1 bit for channel 1. When read, this bit indicates the OCU bit pin value. Output is inverted before module output if OCUn_OCMCR01:INV1 = '1'. When Output Compare Unit channel 1 is in debug mode, this bit is forced to '0' if OCUn_OCMCR01:INV1 = '0' else this bit is forced to '1'. '0': is returned when the OCU output signal is set to '0' and OCUn_OCMCR01:INV0 = '0' or the OCU output signal is set to `1' and OCUn_OCMCR01:INV1 = '1'. '1': is returned when the OCU output signal is set to '1' and OCUn_OCMCR01:INV1 = '0' or the OCU output signal is set to '0' and OCUn_OCMCR01:INV1 = '1'
[8]	OTD0	Output Pin Level Select for Channel 0 The initial value of the OCU output pin is '0'. This bit can be changed only when the compare operation is disabled by configuring the OCUn_OCS01:CST0 bit for channel 0. When read, this bit indicates the OCU bit pin value. Output is inverted before module output if OCUn_OCMCR01:INV0 = '1'. When Output Compare Unit channel 0 is in debug mode, this bit is forced to '0' if OCUn_OCMCR01:INV0 = '0' else this bit is forced to '1'. '0': is returned when the OCU output signal is set to '0' and OCUn_OCMCR01:INV0 = '0' or the OCU output signal is set to `1' and OCUn_OCMCR01:INV1 = '1'. '1': is returned when the OCU output signal is set to '1' and OCUn_OCMCR01:INV1 = '0' or the OCU output signal is set to '0' and OCUn_OCMCR01:INV1 = '1'.
[7:6]	read0	-
[5]	ICE1	Compare Interrupt Enable for Channel 1 This bit is used as an output compare interrupt enable bit. When the compare match status bit OCUn_OSR01:ICP1 is set and the corresponding interrupt enable bit ICE1 is set an output compare interrupt occurs. '0': Disables output compare interrupt '1': Enables output compare interrupt
[4]	ICE0	Compare Interrupt Enable for Channel 0 This bit is used as an output compare interrupt enable bit. When the compare match status bit OCUn_OSR01:ICP0 is set and the corresponding interrupt enable bit ICE0 is set an output compare interrupt occurs. '0': Disables output compare interrupt '1': Enables output compare interrupt
[3:2]	read0	-

Table 26-18. Control Register (OCUn_OCS01) bits

Bit Position	Bit Field Name	Bit Description
[1]	CST1	Comparison with Timer for Channel 1 This bit is used to enable the compare operation. '0': Compare operation disabled '1': Compare operation enabled **Note:** Ensure that a value is written to the Output Compare Register before the compare operation is enabled. Since output compare is synchronized with the 16-bit FRT clock stopping the 16-bit FRT stops the compare operation.
[0]	CST0	Comparison with Timer for Channel 0 This bit is used to enable the compare operation. '0': Compare operation disabled '1': Compare operation enabled **Note:** Ensure that a value is written to the Output Compare Register before the compare operation is enabled. Since output compare is synchronized with the 16-bit Free Running Timer clock stopping the 16-bit Free Running Timer stops the compare operation.

26.3.1.5 Control Clear Register (OCUn_OCSC01)

The Control Clear Register (OCUn_OCSC01) allows the clearing of bits within the control register (OCUn_OCS01).

Control Clear Register (OCUn_OCSC01)

Figure 26-22. Control Clear Register (OCUn_OCSC01)

OCUn_OCSC01															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	OTDC1	OTDC0	read0	read0	ICEC1	ICEC0	read0	read0	CSTC1	CSTC0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-19. Control Clear Register (OCUn_OCSC01) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	read0	-
[9]	OTDC1	Output Pin Level Select Clear for Channel 1 '0': No effect '1': Clears the output data register before selectable output inversion Reading this bit always returns '0'.
[8]	OTDC0	Output Pin Level Select Clear for Channel 0 '0': No effect '1': Clears the output data register before selectable output inversion Reading this bit always returns '0'.
[7:6]	read0	-
[5]	ICEC1	Compare Interrupt Enable Clear for Channel 1 '0': No effect '1': Clears OCUn_OCS01:ICE1 Reading this bit always returns '0'.
[4]	ICEC0	Compare Interrupt Enable Clear for Channel 0 '0': No effect '1': Clears OCUn_OCS01:ICE0 Reading this bit always returns '0'.
[3:2]	read0	-

Table 26-19. Control Clear Register (OCUn_OCSC01) bits

Bit Position	Bit Field Name	Bit Description
[1]	CSTC1	Comparison with Timer Clear for Channel 1 '0': No effect '1': Clears OCUn_OCS01:CST1 Reading this bit always returns '0.'
[0]	CSTC0	Comparison with Timer Clear for Channel 0 '0': No effect '1': Clears OCUn_OCS01:CST0 Reading this bit always returns '0'.

26.3.1.6 Control Set Register (OCUn_OCSS01)

The Control Set Register (OCUn_OCSS01) allows the setting of bits within the Control Register (OCUn_OCS01).

Control Set Register (OCUn_OCSS01)

Figure 26-23. Control Set Register (OCUn_OCSS01)

OCUn_OCSS01															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	OTDS1	OTDS0	read0	read0	ICES1	ICES0	read0	read0	CSTS1	CSTS0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-20. Control Set Register (OCUn_OCSS01) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	read0	-
[9]	OTDS1	Output Pin Level Select Set for Channel 1 '0': No effect '1': Sets the output data register before selectable output inversion Reading this bit always returns '0'.
[8]	OTDS0	Output Pin Level Select Set for Channel 0 '0': No effect '1': Sets the output data register before selectable output inversion Reading this bit always returns '0'.
[7:6]	read0	-
[5]	ICES1	Compare Interrupt Enable Set for Channel 1 '0': No effect '1': Sets OCUn_OCS01:ICE1 Reading this bit always returns '0'.
[4]	ICES0	Compare Interrupt Enable Set for Channel 0 '0': No effect '1': Sets OCUn_OCS01:ICE0 Reading this bit always returns '0'.
[3:2]	read0	-

Table 26-20. Control Set Register (OCUn_OCSS01) bits

Bit Position	Bit Field Name	Bit Description
[1]	CSTS1	Comparison with Timer Set for Channel 1 '0': No effect '1': Sets OCUn_OCS01:CST1 Reading this bit always returns '0'.
[0]	CSTS0	Comparison with Timer Set for Channel 0 '0': No effect '1': Sets OCUn_OCS01:CST0 Reading this bit always returns '0'.

26.3.1.7 Status Register (OCUn_OSR01)

The Status Register (OCUn_OSR01) shows the compare match status.

Status Register (OCUn_OSR01)

Figure 26-24. Status Register (OCUn_OSR01)

OCUn_OSR01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	ICP1	ICP0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp
0	0	0	0	0	0	0	0

Table 26-21. Status Register (OCUn_OSR01) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	ICP1	Compare Match Status for Channel 1 When the Output Compare Register value matches the 16-bit FRT value this bit is set to '1'. When a compare match status bit ICP1 is set and the corresponding interrupt enable bit OCUn_OCS01:ICE1 is enabled an output compare interrupt occurs. '0': No compare match '1': Compare match ICP1 set has higher priority over clear if both events occur simultaneously.
[0]	ICP0	Compare Match Status for Channel 0 When the Output Compare Register value matches the 16-bit FRT value this bit is set to '1'. When a compare match status bit ICP0 is set and the corresponding interrupt enable bit OCUn_OCS01:ICE0 is enabled an output compare interrupt occurs. '0': No compare match '1': Compare match ICP0 set has higher priority over clear if both events occur simultaneously.

26.3.1.8 Status Clear Register (OCUn_OSCR01)

The Status Clear Register (OCUn_OSCR01) allows the clearing of bits within the status register (OCUn_OSR01).

Status Clear Register (OCUn_OSCR01)

Figure 26-25. Status Clear Register (OCUn_OSCR01)

OCUn_OSCR01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	ICPC1	ICPC0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 26-22. Status Clear Register (OCUn_OSCR01) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	ICPC1	Compare Match Status Clear for Channel 1 '0': No effect '1': Clears OCUn_OSR01:ICP1 Reading this bit always returns '0'.
[0]	ICPC0	Compare Match Status Clear for Channel 0 '0': No effect '1': Clears OCUn_OSR01:ICP0 Reading this bit always returns '0'.

26.3.1.9 Extended Output Compare Control Status Register(OCUn_EOCS01)

The Extended Output Compare Control Status Register (OCUn_EOCS01) includes the buffer enable/disable function, buffer transfer selection function, fixed mode selection function and the data value of the fixed mode.

Extended Output Compare Control Status Register (OCUn_EOCS01)

Figure 26-26. Extended Output Compare Control Status Register (OCUn_EOCS01)

OCUn_EOCS01															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	OFD1	OFM1	read0	read0	OFD0	OFM0	read0	BTS3	BTS2	BUF1	read0	BTS1	BTS0	BUF0
Rp0	Rp0	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-23. Extended Output Compare Control Status Register (OCUn_EOCS01) bits

Bit Position	Bit Field Name	Bit Description
[15:14]	read0	-
[13]	OFD1	Output Fixed Data for Channel 1 This bit stores the value to which the output pin OUT(2n+1) is to be set during fixed mode of operation OCUn_EOCS01:OFM1 = '1'.
[12]	OFM1	Output Fixed Mode for Channel 1 This bit is used to select between the fixed mode operation and normal operation of the OCU. '0': OCU operates in normal mode '1': OCU operates in fixed mode In normal mode the output pin OUT(2n+1) toggles at each compare event. In fixed mode of operation the output pin OUT(2n+1) will be set to the value of OCUn_EOCS01:OFD1 at each compare event. A compare event can be a compare match or compare clear event in full duty cycle mode, OCUn_OCMCR01:FDEN1 = '1'. If the compare events are disabled, OFM1 and OFD1 have no impact but the bits can be configured. The output pin is than updated at the next compare event. **Note:** When OCU is working in fixed mode the output pin OUT(2n+1) is fixed at a constant value and is independent of the compare match function.
[11:10]	read0	-

Table 26-23. Extended Output Compare Control Status Register (OCUn_EOCS01) bits

Bit Position	Bit Field Name	Bit Description
[9]	OFD0	Output Fixed Data for Channel 0 This bit stores the value to which the output pin OUT(2n+0) is to be set during fixed mode of operation OCUn_EOCS01:OFM = '1'.
[8]	OFM0	Output Fixed Mode for Channel 0 This bit is used to select between the fixed mode operation and normal operation of the OCU. '0': OCU operates in normal mode '1': OCU operates in fixed mode In normal mode the output pin OUT(2n+0) toggles at each compare event. In fixed mode the output pin OUT(2n+0) will be set to the value of OCUn_EOCS01:OFD0 at each compare event. A compare event can be a compare match or compare clear event in full duty cycle mode, OCUn_OCMCR01:FDEN0 = '1'. If the compare events are disabled, OFM0 and OFD0 have no impact, but bits can be configured. The output Pin is then updated at the next compare event. **Note:** When OCU is working in fixed mode the output pin OUT(2n+0) is fixed at a constant value and is independent of the compare match function.
[7]	read0	-
[6]	BTS3	Buffer Transfer Selection bit for Channel 1 Based on the value of the BTS bits the buffer register value is transferred to the Output Compare Register OCUn_OCCP1 on different FRT counter events. For details refer Table 26-24.
[5]	BTS2	Buffer Transfer Selection bit for Channel 1. Based on the value of the BTS bits the buffer register value is transferred to the Output Compare Register OCUn_OCCP1 on different FRT counter events. For details refer Table 26-24.
[4]	BUF1	Compare Buffer Enable bit for Channel 1 This bit is used to enable the buffer function for channel 1. '0': Disables the buffer function for channel 1 '1': Enables the buffer function
[3]	read0	-
[2]	BTS1	Buffer Transfer Selection bit for Channel 0 Based on the value of the BTS bits the buffer register value is transferred to the Output Compare Register OCUn_OCCP1 on different FRT counter events. For details refer Table 26-25.
[1]	BTS0	Buffer Transfer Selection bit for Channel 0 Based on the value of the BTS bits the buffer register value is transferred to the Output Compare Register OCUn_OCCP1 on different FRT counter events. For details refer Table 26-25.

Table 26-23. Extended Output Compare Control Status Register (OCUn_EOCS01) bits

Bit Position	Bit Field Name	Bit Description
[0]	BUF0	Compare Buffer Enable bit for Channel 0 This bit is used to enable the buffer function for channel 0. '0': Disables the buffer function for channel 0 '1': Enables the buffer function

Table 26-24. Operation of buffer transfer selection bits (BTS[3:2])

BTS[3]	BTS[2]	Value transferred from OCUn_OCCPB1 to OCUn_OCCP1	Value transferred from OCUn_OCCPBD1 to OCUn_OCCP1
'0'	'0'	At zero detect of FRT	No operation
'0'	'1'	At compare clear of FRT	No operation
'1'	'0'	At zero detect of FRT	At compare clear of FRT
'1'	'1'	Reserved	Reserved

Note: BTS[3:2] = '10' should be used only in up/down count mode of FRT because OCUn_OCCP1 is updated with OCUn_OCCPBD1 after a compare clear event; a zero-trigger event occurs in the next cycle of compare clear event and updates OCUn_OCCP1 with OCUn_OCCPB1.

Table 26-25. Operation of Buffer Transfer Selection bits (BTS[1:0])

BTS[1]	BTS[0]	Value transferred from OCUn_OCCPB0 to OCUn_OCCP0	Value transferred from OCUn_OCCPBD0 to OCUn_OCCP0
'0'	'0'	At zero detect of FRT	No operation
'0'	'1'	At compare clear of FRT	No operation
'1'	'0'	At zero detect of FRT	At compare clear of FRT
'1'	'1'	Reserved	Reserved

Note: BTS[1:0] = '10' should be used only in up/down count mode of FRT because OCUn_OCCP0 is updated with OCUn_OCCPBD0 after a compare clear event; a zero-trigger event occurs in the next cycle of compare clear event and updates OCUn_OCCP0 with OCUn_OCCPB0.

Table 26-26. Relationship between the bits OFM, CST and OTD

OCUn_EOCS01 :OFMx	OCUn_OCS01 :CSTx	OCUn_OCS01 :OTDx	OUTx	Explanation
'0'	'0'	'0'	'0'	Normal mode (OCUn_EOCS01:OFMx = '0') Compare match disabled (OCUn_OCS01:CSTx = '0') The OCUn_OCS01:OTDx value is updated to the OUTx pin

Table 26-26. Relationship between the bits OFM, CST and OTD

OCUn_ EOCS01 :OFMx	OCUn_ OCS01 :CSTx	OCUn_ OCS01 :OTDx	OUTx	Explanation
'0'	'0'	'1'	'1'	Normal mode (OCUn_EOCS01:OFMx = '0') Compare match disabled (OCUn_OCS01:CSTx = '0') The OCUn_OCS01:OTDx value is updated to the OUTx pin
'0'	'1'	X	Toggles	Normal mode (OCUn_EOCS01:OFMx = '0') Compare match enabled (OCUn_OCS01:CSTx = '1') OCUn_OCS01:OTDx does not have any affect on OUTx OUTx toggles on compare match event and other conditions like full duty range, CMOD,..etc. For more details refer to 26.3.2 Operation of 16-bit Output Compare Unit. Note: The initial value of OCUn_OCS01:OTDx is the initial value for OUTx.
'1'	'0'	X	OFDx	Fixed mode(OCUn_EOCS01:OFMx = '1') Compare match disabled(OCUn_OCS01:CSTx = '0') OUTx will be updated with OCUn_EOCS01:OFDx value only if OCUn_OCMCR01:FDENx = '1' and Compare clear event from FRT occurs If OCUn_OCMCR01:FDENx = '0' or compare clear event from FRT is not present, then there will be no event to update OUTx During the condition (OCUn_OCS01:CSTx = '0') OCUn_EOCS01:OFMx/ OFDx has no impact, but bits can be configured For more details refer to Figure 26-43. Note: The initial value of OCUn_OCS01:OTDx is the initial value for OUTx.
'1'	'1'	X	OFDx	Fixed mode(OCUn_EOCS01:OFMx = '1'): Compare match enabled (OCUn_OCS01:CSTx = '1') OUTx will be updated on compare match event by OCUn_EOCS01:OFDx and also by compare clear event If OCUn_OCMCR01:FDENx = '1', OCUn_OCS01:OTDx does not have any affect For more details refer to Figure 26-43. Note: The initial value of OCUn_OCS01:OTDx is the initial value for OUTx.

Note:

'x' can be '0' or '1' signifying two channels of a single module of OCU.

'X' indicates don’t care.

26.3.1.10 Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01)

The Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01) allows the setting of bits within the Extended Output Compare Control Status Register (OCUn_EOCS01).

Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01)

Figure 26-27. Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01)

OCUn_EOCSSH01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	OFDS1	OFMS1	read0	read0	OFDS0	OFMS0
Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 26-27. Extended Output Compare Control Status Set Register High (OCUn_EOCSSH01) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5]	OFDS1	Output Fixed Data Set bit for Channel 1 '0': No effect '1': Sets OCUn_EOCS01:OFD1 Reading this bit always returns '0'.
[4]	OFMS1	Output Fixed Mode Set bit for Channel 1 '0': No effect '1': Sets OCUn_EOCS01:OFM1 Reading this bit always returns '0'.
[3:2]	read0	-
[1]	OFDS0	Output Fixed Data Set bit for Channel 0 '0': No effect '1': Sets OCUn_EOCS01:OFD0 Reading this bit always returns '0'.
[0]	OFMS0	Output Fixed Mode Set bit for Channel 0 '0': No effect '1': Sets OCUn_EOCS01:OFM0 Reading this bit always returns '0'.

26.3.1.11 Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01)

The Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01) contains the bits to clear the bits of the Extended Output Compare Control Status Register High (OCUn_EOCSH01).

Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01)

Figure 26-28. Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01)

OCUn_EOCSCH01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	OFDC1	OFMC1	read0	read0	OFDC0	OFMC0
Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 26-28. Extended Output Compare Control Status Clear Register High (OCUn_EOCSCH01) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5]	OFDC1	Output Fixed Data Clear bit for Channel 1 '0': No effect '1': Clears OCUn_EOCS01:OFD1 Reading this bit always returns '0'.
[4]	OFMC1	Output Fixed Mode Clear bit for Channel 1 '0': No effect '1': Clears OCUn_EOCS01:OFM1 Reading this bit always returns '0'.
[3:2]	read0	-
[1]	OFDC0	Output Fixed Data Clear bit for Channel 0 '0': No effect '1': Clears OCUn_EOCS01:OFD0 Reading this bit always returns '0'.
[0]	OFMC0	Output Fixed Mode Clear bit for Channel 0 '0': No effect '1': Clears OCUn_EOCS01:OFM0 Reading this bit always returns '0'.

26.3.1.12 Debug Configuration Register (OCUn_DEBUG01)

The Debug Configuration Register (OCUn_DEBUG01) can be configured by the CPU to enable or disable the debug mode for the corresponding OCU channels.

Debug Configuration Register (OCUn_DEBUG01)

Figure 26-29. Debug Configuration Register (OCUn_DEBUG01)

OCUn_DEBUG01							
read0	read0	read0	read0	read0	read0	DBGEN1	DBGEN0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 26-29. Debug Configuration Register (OCUn_DEBUG01) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	DBGEN1	Enable Debug Mode for Channel 1 This bit is used to enable/disable the debug mode support for channel 1. '0': OCU channel 1 debug mode support is disabled '1': OCU channel 1 debug mode support is disabled When DBGEN1 is set to '1' and the processor is in debug state,the toggling of the output signal OUT(2n+1) is stopped. For the definition of the debug state refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[0]	DBGEN0	Enable Debug Mode for Channel 0 This bit is used to enable/disable the debug mode support for channel 0. '0': OCU channel 0 debug mode support is disabled '1': OCU channel 0 debug mode support is enabled When DBGEN0 is set to '1' and the processor is in debug state, the toggling of the output signal OUT(2n+0) is stopped. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.

26.3.1.13 DMA Configuration Register (OCUn_DMACFG01)

The DMA Configuration Register (OCUn_DMACFG01) can be configured by the CPU to enable or disable the DMA transfer for the corresponding OCU channels.

DMA Configuration Register (OCUn_DMACFG01)

Figure 26-30. DMA Configuration Register (OCUn_DMACFG01)

OCUn_DMACFG01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	ENDMAREQ1	ENDMAREQ0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 26-30. DMA Configuration Register (OCUn_DMACFG01) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	ENDMAREQ1	Enable DMA Request for Channel 1 This bit is used to enable/disable DMA request for channel 1. '0': OCU channel 1 DMA request is disabled '1': OCU channel 1 DMA request is enabled
[0]	ENDMAREQ0	Enable DMA Request for Channel 0 This bit is used to enable/disable DMA request for channel 0. '0': OCU channel 0 DMA request is disabled '1': OCU channel 0 DMA request is enabled

26.3.1.14 Compare Mode Control Register (OCUn_OCMCR01)

The Compare Mode Control Register (OCUn_OCMCR01) includes the compare match output setting function, function for inverting the output channel and function for enabling the full duty range for the output.

Compare Mode Control Register (OCUn_OCMCR01)

Figure 26-31. Compare Mode Control Register (OCUn_OCMCR01)

OCUn_OCMCR01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	FDEN1	INV1	CMPMD1	read0	FDEN0	INV0	CMPMD0
Rp0	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 26-31. Compare Mode Control Register (OCUn_OCMCR01) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	FDEN1	Full Duty Range Enable bit for Channel 1 Full Duty Range signifies the generation of PWM signals having a wavelength of full duty range. '0': Full duty range feature is disabled The output channel 1 toggles when the Output Compare Register (OCUn_OCCP1) matches the value of the FRT. '1': Full duty range feature is enabled **Note:** For more details on the behaviour of the output in full duty range under various conditions refer to the Table 26-32.
[5]	INV1	Invert bit for Output of Channel 1 This bit is used to invert the output value of OCU output pins. '0': Output value of the OCU output pin is not inverted '1': Output value of the OCU output pin is inverted

Table 26-31. Compare Mode Control Register (OCUn_OCMCR01) bits

Bit Position	Bit Field Name	Bit Description
[4]	CMPMD1	Compare Match Output Setting bit for Channel 1 This bit is used to change the pin output level immediately after a match occurs. This function can only be used if OCUn_OCS01:CMOD[1:0] = '00' and OCUn_OCMCR01:FDEN = '1'. CMPMD1='0': Output value of channel OUT(2n+1)of OCUn will toggle when a match occurs between FRT value and compare register OCUn_OCCP1. CMPMD1='1': Output value of channel OUT(2n+1) of OCU will have the inverted value of the OCUn_OCMCR01:INV1, when a match between FRT value and the compare register OCUn_OCCP1 occurs in the upcount mode. In case of down count mode the channel OUT(2n+1) value will be same as OCUn_OCMCR01:INV1, when a match between OCUn_OCCP1 occurs.
[3]	read0	-
[2]	FDEN0	Full Duty Range Enable bit for Channel 1 Full Duty Range signifies generation of PWM signals having the wave-length of full duty range. '0': Full duty range feature is disabled The output channel 0 toggles when the Output Compare Register (OCUn_OCCP0) matches the value of the Free Running Timer. '1': Full duty range feature is enabled **Note:** For more details on the behaviour of the output in full duty range under various conditions refer to the Table 26-32.
[1]	INV0	Invert bit for Output of Channel 0 This bit is used to invert the output value of Output Compare Unit output pins. '0': Output value of OCU output pin is not inverted '1': Output value of OCU output pin is inverted
[0]	CMPMD0	Compare Match Output Setting bit for Channel 0 This bit is used to change the pin output level immediately after a match occurs. This function can only be used if OCUn_OCS01:CMOD[1:0] = '00' and OCUn_OCMCR01:FDEN = '1'. CMPMD0='0': Output value of channel OUT(2n+0)of OCUn will toggle when a match occurs between the FRT value and the compare register OCUn_OCCP0. CMPMD0='1': Output value of channel OUT(2n+0) of OCU will have the inverted value of the OCUn_OCMCR01:INV0, when a match between the FRT value and the compare register OCUn_OCCP0 occurs in the upcount mode. In case of down count mode the channel OUT(2n+1) value will be same as OCUn_OCMCR01:INV0, when a match between OCUn_OCCP1 occurs.

Table 26-32. Output behavior depending on OCUn_OCCP value in full duty range

OCUn_OCCP values under various conditions	Output behavior in full duty range
OCUn_OCCP0/OCUn_OCCP1 = 0x0000	Output OUT (2n+0)/ OUT (2n+1) is independent of FRT counter value and it is equal to the value of OCUn_OCMCR01:INV1. For more details refer to Figure 26-40.
FRTn_CPCLR > OCUn_OCCP1/OCUn_OCCP0 > 0x0000	Output OUT(2n+0)/OUT(2n+1) is dependent on OCUn_OCMCR01:CMPMD1. The output behavior depending on the value of CMPMD bits are described in the description of CMPCMD bits.
OCUn_OCCP0/OCUn_OCCP1 ≥ FRTn_CPCLR	Output OUT(2n+1) is dependent on an internal compare match flag.

Notes:

1. The internal compare match flag is set at compare match event and reset at compare clear event. The Output of OCU does not change if the compare match flag is set. The output OUT(2n+0)/ OUT(2n+1) becomes equal to the inverse of the OCUn_OCMR01:INV1 bit value whenever a compare match occurs and internal compare match flag is '0'.
2. For more details refer to Figure 26-41 and Figure 26-42.

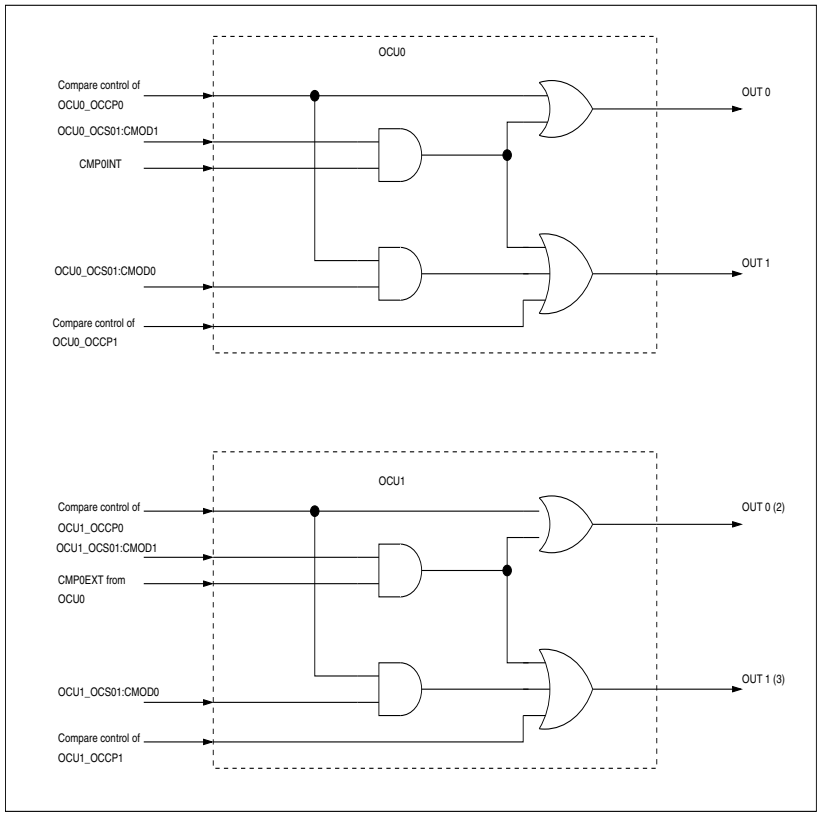
26.3.2 Operation of 16-bit Output Compare Unit

In the 16-bit Output Compare operation, an interrupt request flag can be set and the output level can be toggled when the specified output compare register value matches the 16-bit Free Running Timer value. The OCUn_OCS01:CMOD0 and OCUn_OCS01:CMOD1 bits can be used to define the corresponding compare registers for each pin.

Generation of three PWM signals by using four OCU channels

The following scheme can be used to generate three PWM signals by using four OCU channels. Output Compare Unit 0 and 1 can be combined in order to synchronize output channel 0 to 3.

Figure 26-32. Block diagram of output selection of OCU1



For OCU module 0 the output OUT1 (channel 1) toggles relative to OUT0 (channel 0) if CMOD[1:0] = '01' if there is a match with OCU0_OCCP0. For OCU module 1, the comparison result from OCU0 channel 0 is carried inside OCU1 by CMP0EXT. Depending on the value of OCU1_OCS01:CMOD[1:0] = '10' the output signals of OCU1 will toggle relative to the output OUT0 of OCU1 if there is a match with OCU0_OCCP0. For more details refer to Table 26-33.

Table 26-33. Functions of CMOD1 and CMOD0 bits

CMOD bits value		Description	
OCU(2n+0)_OCS01		Toggle upon match of OCU(2n+0)_OCCP0/ OCU(2n+0)_OCCP1	
CMOD1	CMOD0	OUT(4n+0) toggle on match with the listed OCCP numbers	OUT(4n+1) toggles on match with the listed OCCP numbers
x	'0'	OCU(2n+0)_OCCP0	OCU(2n+0)_OCCP1

Table 26-33. Functions of CMOD1 and CMOD0 bits

CMOD bits value		Description	
OCU(2n+0)_OCS01		Toggle upon match of OCU(2n+0)_OCCP0/ OCU(2n+0)_OCCP1	
x	'1'	OCU(2n+0)_OCCP0	OCU(2n+0)_OCCP0, OCU(2n+0)_OCCP1
OCU(2n+1)_OCS01		Toggle upon match of OCU(2n+1)_OCCP0/ OCU(2n+1)_OCCP1	
CMOD1	CMOD0	OUT(4n+2) toggles on match with the listed OCCP numbers	OUT(4n+3) toggles on match with the listed OCCP numbers
'0'	'0'	OCU(2n+1)_OCCP0	OCU(2n+1)_OCCP1
'0'	'1'	OCU(2n+1)_OCCP0	OCU(2n+1)_OCCP0, OCU(2n+1)_OCCP1
'1'	'0'	OCU(2n+0)_OCCP0, OCU(2n+1)_OCCP0	OCU(2n+0)_OCCP0, OCU(2n+1)_OCCP1
'1'	'1'	OCU(2n+0)_OCCP0, OCU(2n+1)_OCCP0	OCU(2n+0)_OCCP0, OCU(2n+1)_OCCP0, OCU(2n+1)_OCCP1
Note: 'n' indicates the OCU channel number.			

Note: For more details on the generation of three PWM signals by using four OCU channels refer to Figure 26-35.

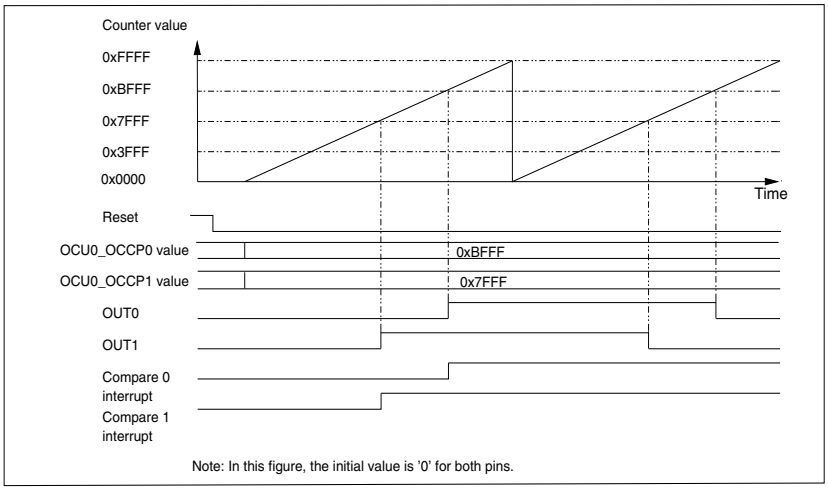
Sample output waveform when OCU0_OCS01:CMOD[1:0] = '00'

When OCUn_OCS01:CMOD[1:0] = '00', the output level of the pin corresponding to the output compare register is reversed on every match with the FRT counter value. Every channel output value is controlled by one output compare register.

OUT0: The level is only reversed by a match with the Output Compare Register 0 (OCU0_OCCP0).

OUT1: The level is only reversed by a match with the Output Compare Register 1 (OCU0_OCCP1).

Figure 26-33. Sample of the output waveform when OCU0_OCS01:CMOD[1:0] = '00'



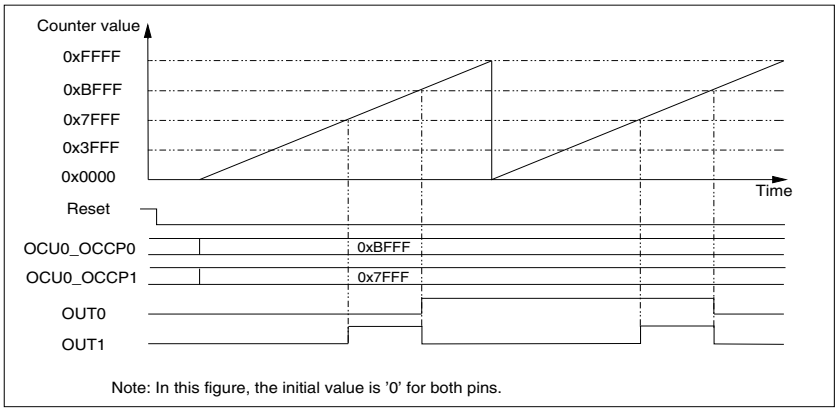
Sample output waveform with two compare registers when OCU0_OCS01:CMOD[1:0] = '01'

When OCU0_OCS01:CMOD[1:0] = '01', the output level of the pin corresponding to the Output Compare Register 0 (OCU0_OCCP0) is reversed upon every match with the OCU0_OCCP0 register value. This is identical to the behavior for OCU0_OCS01:CMOD[1:0] = '00'. However, the output level of the OUT1 is reversed upon a match with either Output Compare Register 0 (OCU0_OCCP0) or Output Compare Register 1 (OCU0_OCCP1). This allows tthe definition of a pulsed signal with one edge defined by the value in the Output Compare Register 0 (OCU0_OCCP0) and the other edge defined by Output Compare Register 1 (OCU0_OCCP1) or vice versa. If both Output Compare Registers have the same value, the operation is identical to the case for OCU0_OCS01:CMOD[1:0] = '00'.

A pulse width modulated signal with differing frequency can be defined by using this mode together with the reset option of the Output Compare Register match for the FRT (FRTn_TCCS: MODE bit). OUT0 (2): The level is only reversed by a match with Output Compare Register 0 (OCU0_OCCP0). OUT1 (3): The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0) or with Output Compare Register 1 (OCU0_OCCP1)).

The same behavior is applicable for other out pins of the OCU module 1,2,3.....

Figure 26-34. Sample of an output waveform when OCU0_OCS01:CMOD[1:0] = '01' (no timer reset by match)



Sample output waveform when OCU1_OCS01:CMOD[1:0] = '10'

The operation mode defined by OCU1_OCS01:CMOD[1:0] = '10' is intended for the use of three pulse width modulated signals for each FRT instead of two. If this mode is set to OCU module 1, a match of the FRT value with Output Compare Register 0 (OCU1_OCCP0) reverses both OUT0(2) and OUT1(3) output channels of OCU1. For the third pulsed signal, the OCU0_OCS01:CMOD[1:0] bits of OCU module 0 should be set to '01'.

If register OCU0_OCS01:CMOD[1:0] = '01'

OUT0: The level is only reversed by a match with Output Compare Register 0 (OCU0_OCCP0).

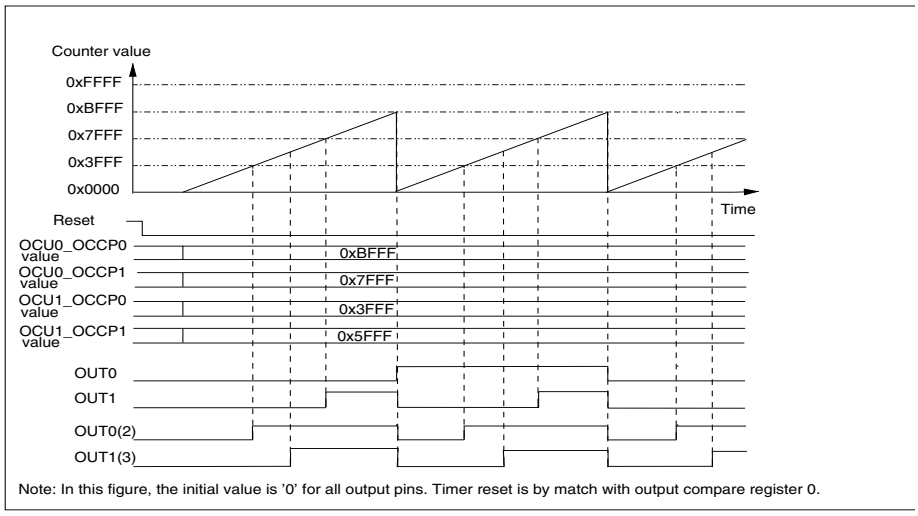
OUT1: The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0) or with Output Compare Register 1 (OCU0_OCCP1).

If register OCU1_OCS01:CMOD[1:0] = '10'

OUT0(2): The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0) or with Output Compare Register 2 (OCU1_OCCP0).

OUT1(3): The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0) or with Output Compare Register 3 (OCU1_OCCP1).

Figure 26-35. Output waveform when OCU0_OCS01:CMOD[1:0] = '01' and OCU1_OCS01:CMOD[1:0] = '10'



Sample output waveform when OCU1_OCS01:CMOD[1:0] = '11'

When OCU1_OCS01:CMOD[1:0] = '11', the output level of the OUT1(3) pin is reversed by the Output Compare Registers 0 (OCU0_OCCP0), 2 (OCU1_OCCP0) or 3 (OCU1_OCCP1). For the pin OUT(2n+1), this setting is identical to OCU0_OCS01:CMOD[1:0] = '01' (for more details refer to Table 26-33).

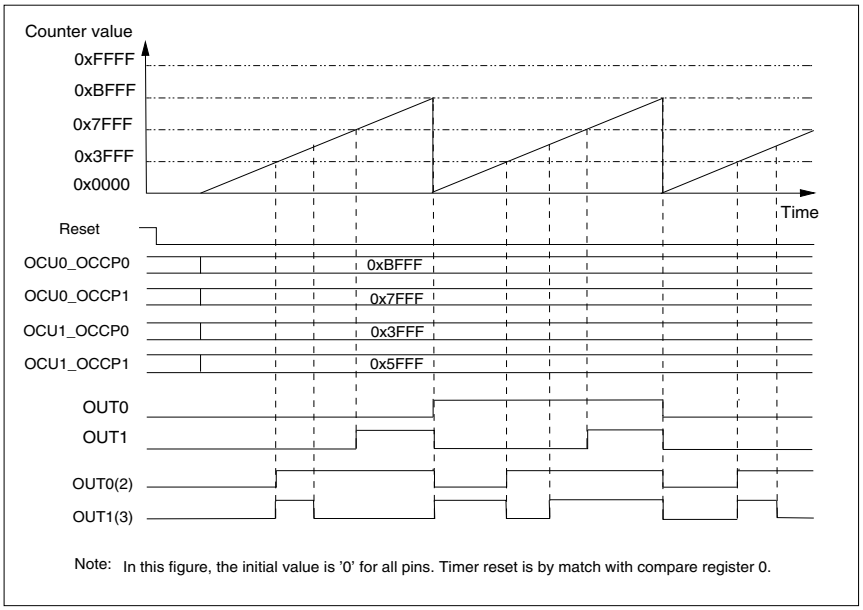
OUT0: The level is only reversed by a match with Output Compare Register 0 (OCU0_OCCP0).

OUT1: The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0) or with Output Compare Register 1 (OCU0_OCCP1).

OUT0(2): The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0) or with Output Compare Register 2 (OCU1_OCCP0).

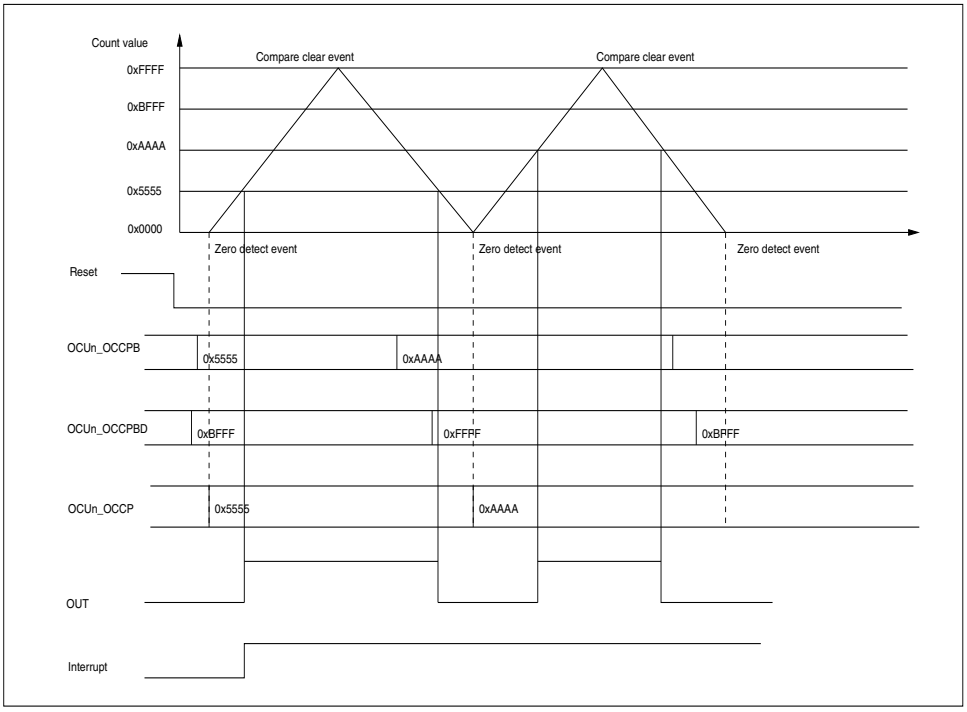
OUT1(3): The level is reversed by a match with Output Compare Register 0 (OCU0_OCCP0), Output Compare Register 2 (OCU1_OCCP0) or with Output Compare Register 3 (OCU1_OCCP1).

Figure 26-36. Output waveform when OCU0_OCS01:CMOD[1:0] = ’11’ and OCU1_OCS01:CMOD[1:0] = ’11’



Sample output waveform when OCUn_OCS01:CMOD[1:0] = ’00', OCUn_EOCS01:BTS[1:0] = ’00’ and OCUn_OCMR01:FDEN[1:0] = ’0'

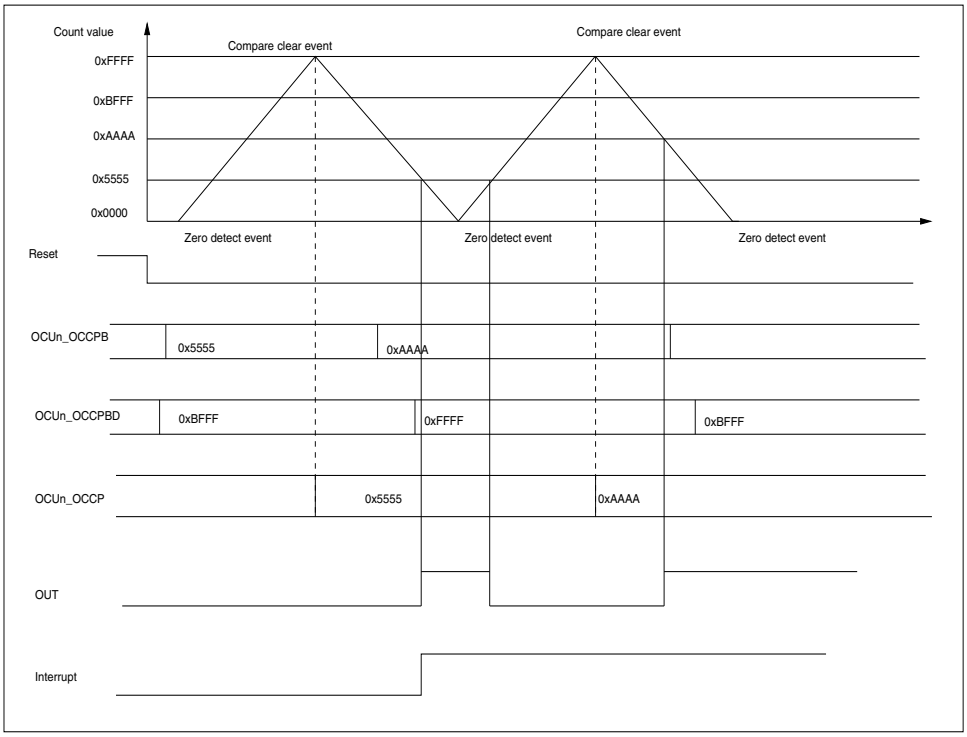
Figure 26-37. Output waveform when OCUn_OCS01:CMOD = ’00’ OCUn_EOCS01:BTS[1:0] = ’00', and OCUn_OCMR01:FDEN = ’0'



The above waveform shows the transfer of the values present in the Output Compare Buffer Register to the Output Compare Register on events which are determined by the OCUn_EOCS01:BTS[1:0] bits. As OCUn_EOCS01:BTS[1:0] = ’00’, OCUn_OCCPB data will be transferred to the OCUn_OCCP register in the event of a zero detect event of the FRT counter. For more details refer Table 26-25.

Sample output waveform when OCUn_OCS01:CMOD[1:0] = ’00', OCUn_OCMR01:FDEN = ’0', and OCUn_EOCS01:BTS[1:0] = ’01’

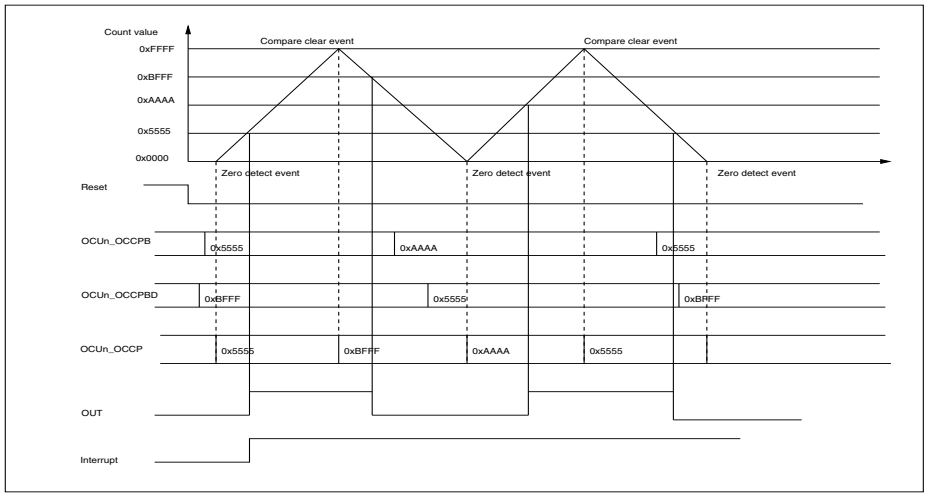
Figure 26-38. Output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_OCMR01:FDEN = ’0’, and OCUn_EOCS01:BTS[1:0] = ’01’



The above waveform shows the transfer of the values present in the Output Compare Buffer Register to the Output Compare Register on events which are determined by the OCUn_EOCS01:BTS[1:0] bits. As OCUn_EOCS01:BTS[1:0] = ’01’, OCUn_OCCPB data will be transferred to the OCUn_OCCP register on the event of a compare clear event of the FRT counter. For more details refer Table 26-25.

Sample output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCMR01:FDEN = ’0’

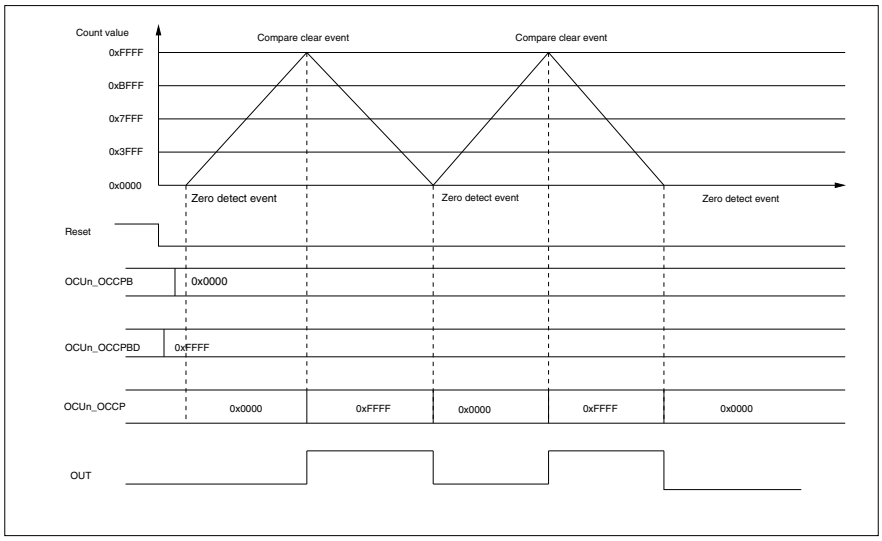
Figure 26-39. Output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCMR01:FDEN = ’0’



The above waveform shows the transfer of the values present in the Output Compare Buffer Register to the Output Compare Register on events which are determined by the OCUn_EOCS01:BTS[1:0] bits. As OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCCPB data will be transferred to the OCUn_OCCP register in the event of a zero detect of the FRT counter, and OCUn_OCCPBD data will be transferred to OCUn_OCCP in the event of a compare clear event of the FRT counter. For more details refer Table 26-25.

Sample output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCMR01:FDEN = ’1’, OCUn_OCCP01:OCCP = 0x0000/0xFFFF

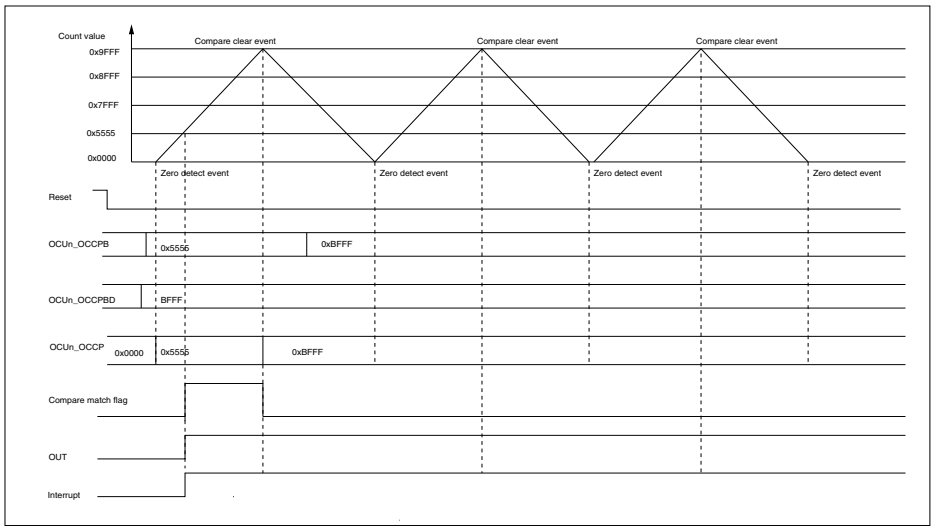
Figure 26-40. Output waveform when OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCMR01:FDEN = ’1’ and OCUn_OCCP01:OCCP = 0x0000/0xFFFF



The above waveform shows a full duty enable condition. As the value of OCUn_OCCP toggles between 0x0000/0xFFFF because of OCUn_EOCS01:BTS[1:0]= '10', the output channel (OUT) will toggle alternately at every compare clear event and zero detect event. The toggling is because the OCUn_OCCP value chnages.

Sample output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_OCMR01:FDEN = ’1’, OCUn_OCMR01:INV = ’0’, OCUn_EOCS01:BTS[1:0] = ’10’ and FRTn_CPCLR < OCUn_OCCP01:OCCP

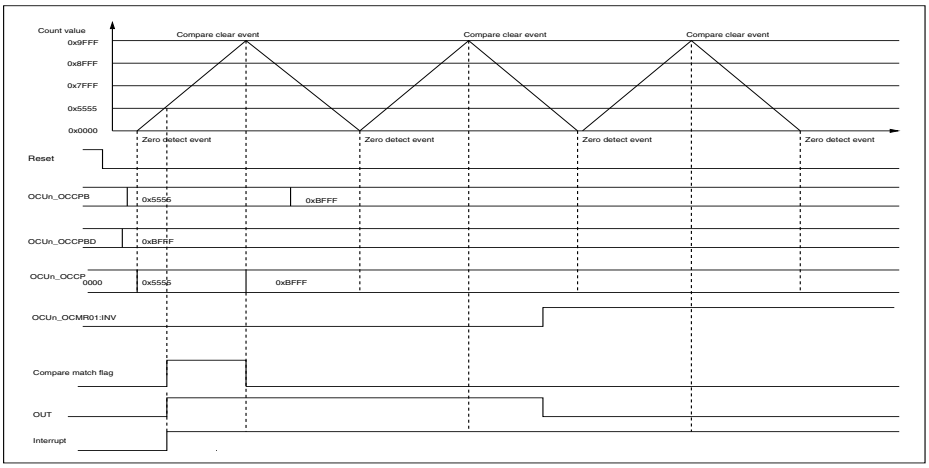
Figure 26-41. Output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_EOCS01:BTS = ’10’, OCUn_OCMR01:FDEN = ’1’ and OCUn_OCMR01:INV = ’0’



The above condition shows the output behavior when full duty range is enabled OCUn_OCMCR01:FDEN = ’1’ and the OCUn_OCCP01:OCCP value is less than FRTn_CPCLR. The waveform shows the internal compare match flag which goes high at a compare match event and is reset at a compare clear event. The output becomes equal to the inverse of OCUn_OCMCR01:INV bit value whenever a compare match event occurs.

Sample output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_OCMR01:FDEN = ’1’, OCUn_OCMR01:INV = changed from ’0’ to ’1’ and OCUn_EOCS01:BTS[1:0] = ’10’ and FRTn_CPCLR < OCUn_OCCP01:OCCP

Figure 26-42. Output waveform when OCUn_OCS01:CMOD[1:0] = ’00’, OCUn_EOCS01:BTS = ’10’, OCUn_OCMR01:FDEN = ’1’, and OCUn_OCMR01:INV = ’0’ -> ’1’



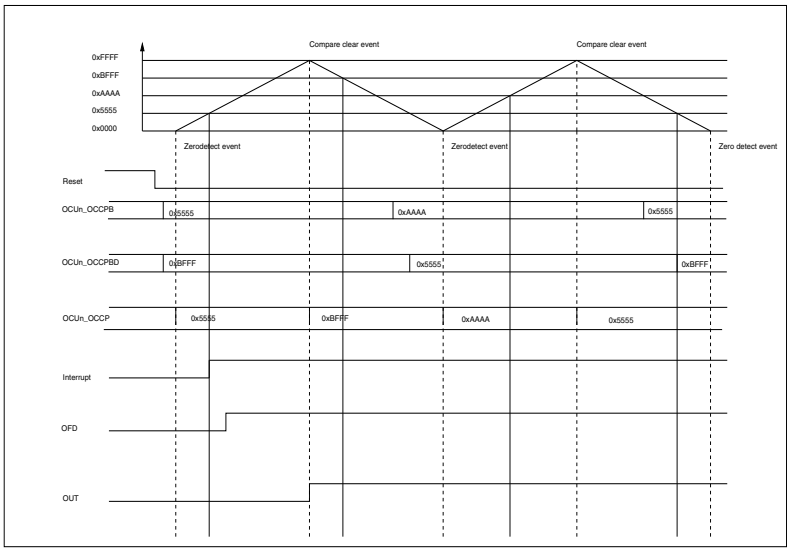
The above condition shows the output behavior when full duty range is enabled OCUn_OCMCR01:FDEN = ’1’ and the OCUn_OCCP01:OCCP value is less than FRTn_CPCLR. The waveform shows the internal compare match flag which goes high at compare match event and is reset at a clear event. The output becomes equal to the inverse of OCUn_OCMCR01:INV bit value whenever a compare match event occurs. In this case initially OCUn_OCMCR01:INV = ’0’ hence channel output OUT becomes high and then when OCUn_OCMCR01:INV becomes ’1’ the output settles down to ’0’.

Sample output waveform when OCUn_OCS01:CMOD = ’00’, OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCMR01:INV = ’0’, OCUn_OCMR01:FDEN = ’1’, and OCUn_EOCS01:OFM = ’1’

When OCUn_EOCS01:OFM0, OFM1 = '1' then the fixed mode output is selected. In this mode the output of the OCU is fixed to the value configured in OCUn_EOCS01:OFD01 and the output becomes equal to OFD on either a compare clear match or compare clear event from the FRT. A compare clear event is applicable only when OCUn_OCMR01:FDEN = '1'.

Note: In fixed mode the compare match operation of the OCU is not stopped and hence an interrupt is generated; the output however does not toggle.

Figure 26-43. Output waveform when OCUn_OCS01:CMOD = ’00’, OCUn_EOCS01:BTS[1:0] = ’10’, OCUn_OCMR01:INV= ’0’, OCUn_OCMR01:FDEN = ’1’ and OCUn_EOCS01:OFM = ’1’



Output compare timing

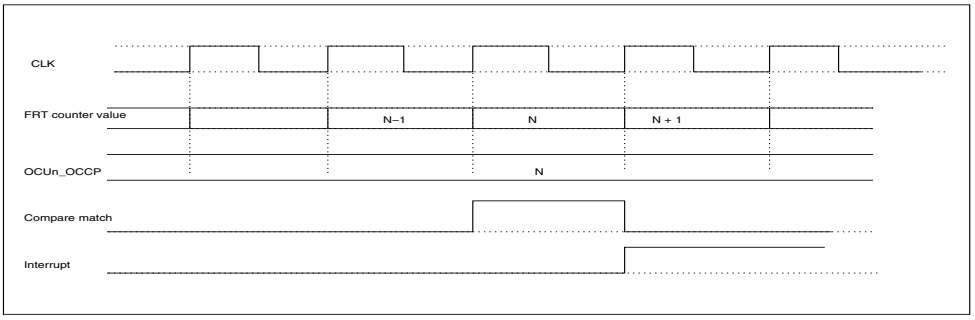
In an output compare operation, a compare match signal is generated when the FRT value matches the specified output compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing in the event of a compare match is synchronized with the counter timing.

Compare operation in the event of an update of the output compare register

When the output compare register is updated, a comparison with the counter value is not performed.

Interrupt timing

Figure 26-44. Interrupt timing

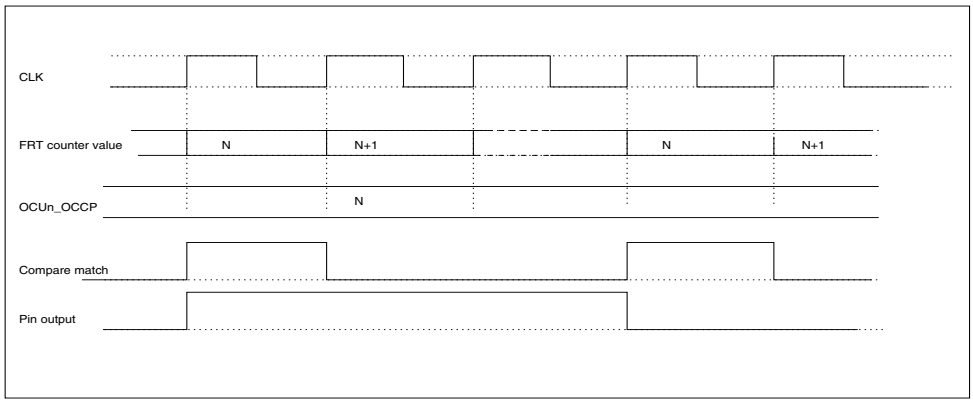


The diagram above shows the timing point where the interrupt is generated with respect to the FRT counter value, the clock and the compare match event. N-1, N and N+1 depicts instances of the FRT counter value before the compare match event has occurred, at the time of the compare match event and after the compare match event has occurred respectively. The diagram shows that interrupt is generated at the point after the compare match event has occurred.

Output pin change timing

Figure 26-45 shows the behavior of the pin output with the default configuration of the pin output function.

Figure 26-45. Output pin change timing



The diagram above shows the timing point where the output pin toggles with respect to the FRT counter value, the clock and the compare match event. The output toggles at every compare match event as shown by pin output.

26.4 Input Capture Unit (ICU)

This section describes the features and the block diagram of the ICU.

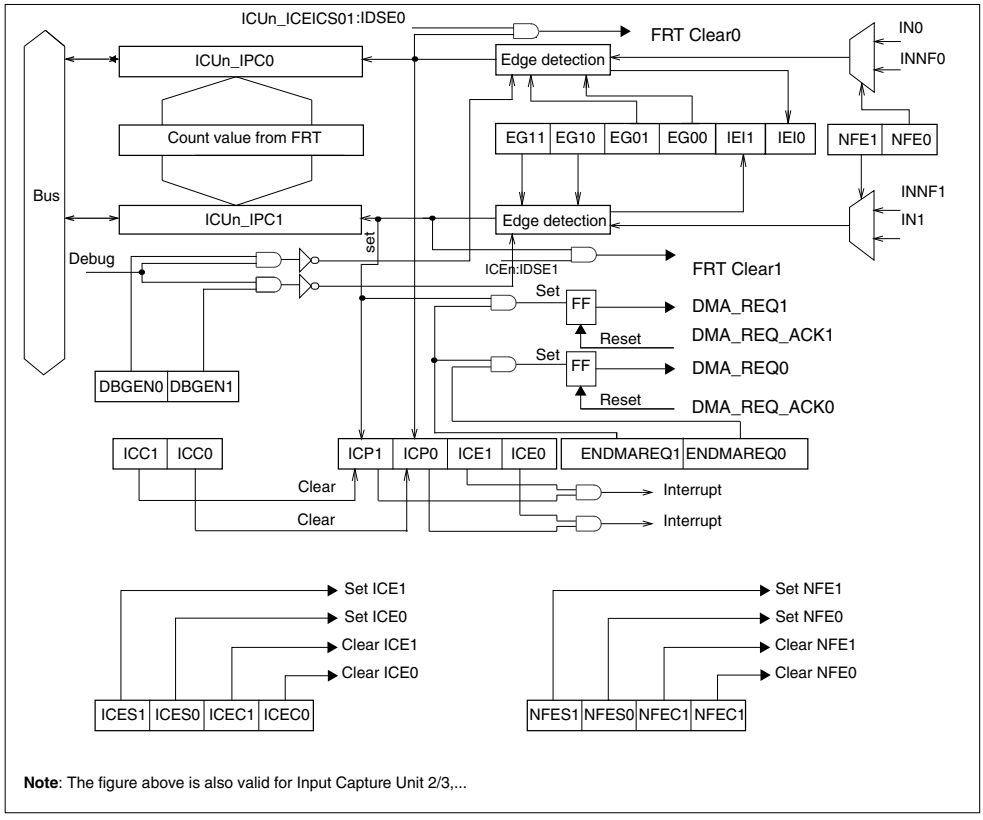
Features of the ICU

The ICU detects rising, falling or both edges of an external input signal and stores the 16-bit Free Running Timer value at that time in a register. In addition, the ICU can generate an interrupt upon detection of an edge. One Input Capture Unit consists of two input capture data registers and one control register. Other features of the ICU include,

- The detected edge of an external input signal can be specified
  - Rising, falling or both edges
- The two input channels can operate independently
- The ICU can clear the FRT at edge detection
- An interrupt can be issued upon a valid edge of an external input signal
  - The DMA can be triggered upon an input capture interrupt
- Support for MCU debug mode

Block diagram of Input Capture Unit

Figure 26-46. Block diagram of the ICU



Note: The figure above is also valid for Input Capture Unit 2/3,...

26.4.1 Input Capture Unit registers

All registers in the ICU module are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of Input Capture Unit

The following registers are available for each instance of the ICU:

- Input Capture Data Register 0 (ICUn_IPC0)
- Input Capture Data Register 1 (ICUn_IPC1)
- Input Capture Clear Register (ICUn_ICC01)
- Input Capture Edge Register and Control Status Register (ICUn_ICEICS01)
- DMA Configuration Register (ICUn_DMACFG01)
- Debug Register (ICUn_DEBUG01)

Memory layout of the ICU registers

Table 26-34. Memory layout of the ICU registers

Offset	+1	+0
0x00000000	ICUn_IPC0 00000000 00000000	
0x00000002	ICUn_IPC1 00000000 00000000	
0x00000004	ICUn_ICC01 00000000 00000000	
0x00000006	ICUn_ICEICS01 00000000 00000000	
0x00000008	ICUn_DEBUG01 00000000	ICUn_DMACFG01 00000000

26.4.1.1 Input Capture Data Register 0 (ICUn_IPC0)

The ICU has a 16-bit Data Register 0 (ICUn_IPC0). This register stores a value from the 16-bit RT, when a valid edge of the corresponding external pin input waveform is detected. For data consistency, it should be accessed only in half word mode only.

Input Capture Data Register 0 (ICUn_IPC0)

Figure 26-47. Input Capture Data Register 0 (ICUn_IPC0)

ICUn_IPC0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
IPC[15]	IPC[14]	IPC[13]	IPC[12]	IPC[11]	IPC[10]	IPC[9]	IPC[8]	IPC[7]	IPC[6]	IPC[5]	IPC[4]	IPC[3]	IPC[2]	IPC[1]	IPC[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-35. Input Capture Data Register 0 (ICUn_IPC0) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	IPC	Input Capture Data Register 0 This register stores the current value from the 16-bit FRT when a valid edge of the corresponding external pin input is detected.

Note: For consistency of data in the upper and lower byte, always read this register in half word mode.

26.4.1.2 Input Capture Data Register 1 (ICUn_IPC1)

The Input Capture Unit has a 16-bit Data Register 1 (ICUn_IPC1). This register stores a value from the 16-bit FRT, when a valid edge of the corresponding external pin input waveform is detected. For data consistency, it should be accessed only in half word mode.

Input Capture Data Register 1 (ICUn_IPC1)

Figure 26-48. Input Capture Data Register 1 (ICUn_IPC1)

ICUn_IPC1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
IPC[15]	IPC[14]	IPC[13]	IPC[12]	IPC[11]	IPC[10]	IPC[9]	IPC[8]	IPC[7]	IPC[6]	IPC[5]	IPC[4]	IPC[3]	IPC[2]	IPC[1]	IPC[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-36. Input Capture Data Register 1 (ICUn_IPC1) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	IPC	Input Capture Data Register 1 This register stores the current value from 16-bit FRT when a valid edge of the corresponding external pin input is detected.

Note: For consistency of data in the upper and lower byte, always read this register in half word mode.

26.4.1.3 Input Capture Clear Register (ICUn_ICC01)

The Input Capture Clear Register (ICUn_ICC01) allows the clearing or setting of the bits within the ICUn_ICEICS01 register. Set and clear bits of the same register should not be set at the same time. Set has higher priority over clear if both events occur simultaneously.

Input Capture Clear Register (ICUn_ICC01)

Figure 26-49. Input Capture Clear Register (ICUn_ICC01)

ICUn_ICC01															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	ICES1	ICES0	NFES1	NFES0	read0	read0	ICC1	ICC0	ICEC1	ICEC0	NFEC1	NFEC0	read0	read0
Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-37. Input Capture Clear Register (ICUn_ICC01) bits

Bit Position	Bit Field Name	Bit Description
[15:14]	read0	-
[13]	ICES1	Interrupt Enable Set This bit is used to set Interrupt Enable (ICU channel 1). '0': No effect '1': Sets ICUn_ICEICS01:ICE1 Reading this bit always returns '0'.
[12]	ICES0	Interrupt Enable Set This bit is used to set Interrupt Enable (ICU channel 0). '0': No effect '1': Sets ICUn_ICEICS01:ICE0 Reading this bit always returns '0'.
[11]	NFES1	Noise Filter Enable Set This bit is used to set Noise Filter Enable (ICU channel 1). '0': No effect '1': Sets ICUn_ICEICS01:NFE1 Reading this bit always returns '0'.

Table 26-37. Input Capture Clear Register (ICUn_ICC01) bits

Bit Position	Bit Field Name	Bit Description
[10]	NFES0	Noise Filter Enable Set This bit is used to set Noise Filter Enable (ICU channel 0). '0': No effect '1': Sets ICUn_ICEICS01:NFE0 Reading this bit always returns '0'.
[9:8]	read0	-
[7]	ICC1	Interrupt Request Clear (Input Capture 1) This bit is used as interrupt request clear (ICU channel 1). '0': No effect '1': Clears ICUn_ICEICS01:ICP1 Reading this bit always returns '0'.
[6]	ICC0	Interrupt Request Clear (Input Capture 0) This bit is used as interrupt request clear (ICU channel 0). '0': No effect '1': Clears ICUn_ICEICS01:ICP0 Reading this bit always returns '0'.
[5]	ICEC1	Interrupt Enable Clear This bit is used to clear Interrupt Enable (ICU channel 1). '0': No effect '1': Clears ICUn_ICEICS01:ICE1 Reading this bit always returns '0'.
[4]	ICEC0	Interrupt Enable Clear This bit is used to clear Interrupt Enable (ICU channel 0). '0': No effect '1': Clears ICUn_ICEICS01:ICE0 Reading this bit always returns '0'.
[3]	NFEC1	Noise Filter Enable Clear This bit is used to clear Noise Filter Enable (ICU channel 1). '0': No effect '1': Clears ICUn_ICEICS01:NFE1 Reading this bit always returns '0'.
[2]	NFEC0	Noise Filter Enable Clear This bit is used to clear Noise Filter Enable (ICU channel 0). '0': No effect '1': Clears ICUn_ICEICS01:NFE0 Reading this bit always returns '0'.
[1:0]	read0	-

26.4.1.4 Input Capture Edge Register and Control Status Register (ICUn_ICEICS01)

The ICU is configured by the control status register. It cover the configuration are done for valid edge selection, noise filter enable/disable and interrupt enable/disable.

Input Capture Edge Register and Control Status Register (ICUn_ICEICS01)

Figure 26-50. Input Capture Edge Register and Control Status Register (ICUn_ICEICS01)

ICUn_ICEICS01															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
IDSE1	IDSE0	read0	read0	NFE1	NFE0	IEI1	IEI0	ICP1	ICP0	ICE1	ICE0	EG1[1]	EG1[0]	EG0[1]	EG0[0]
RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	Rp	Rp	Rp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-38. Input Capture Edge Register and Control Status Register (ICUn_ICEICS01) bits

Bit Position	Bit Field Name	Bit Description
[15]	IDSE1	ICU Edge Detection Indication Signal Enable This bit is used to enable the output of ICU edge detection indication signal to the selected FRT (ICU channel 1). '0': ICU edge detection signal is disabled '1': ICU edge detection signal is enabled The ICU edge detection indication signal is used to clear the corresponding FRT on input capture.
[14]	IDSE0	ICU Edge Detection Indication Signal Enable This bit is used to enable the output of the ICU edge detection indication signal to the selected FRT (ICU channel 0). '0': ICU edge detection signal is disabled '1': ICU edge detection signal is enabled The ICU edge detection indication signal is used to clear the corresponding FRT on input capture.
[13:12]	read0	-
[11]	NFE1	Noise Filter Enable This bit is used to enable the noise filter for IN1 input (ICU channel 1). '0': Bypass noise filter '1': Enable noise filter

Table 26-38. Input Capture Edge Register and Control Status Register (ICUn_ICEICS01) bits

Bit Position	Bit Field Name	Bit Description
[10]	NFE0	Noise Filter Enable This bit is used to enable the noise filter for IN0 input (ICU channel 0). '0': Bypass noise filter '1': Enable noise filter
[9]	IEI1	Valid Edge Indication bit for ICU channel 1 This bit is the edge indication bit for the capture register IPC1. '0': Falling edge detected '1': Rising edge detected **Note:** The read value is meaningless, if EG1[1:0] = '00'. The read value is independent from edge selected by EG1[1:0].
[8]	IEI0	Valid Edge Indication bit for ICU channel 0 This bit is the edge indication bit for the capture register IPC0. '0': Falling edge detected '1': Rising edge detected **Note:** The read value is meaningless, if EG0[1:0] = '00'. The read value is independent from edge selected by EG0[1:0].
[7]	ICP1	Interrupt Request Flag This bit is used as the interrupt request flag (ICU channel 1). '0': No edge detected for ICU channel 1 '1': Valid edge detected for ICU channel 1
[6]	ICP0	Interrupt Request Flag This bit is used as the interrupt request flag (ICU channel 0). '0': No edge detected for ICU channel 0 '1': Valid edge detected for ICU channel 0
[5]	ICE1	Interrupt Enable This bit is used to enable the input capture interrupt request (ICU channel 1). '0': ICU channel 1 interrupt is disabled '1': ICU channel 1 interrupt is enabled
[4]	ICE0	Interrupt Enable This bit is used to enable the input capture interrupt request (ICU channel 0). '0': ICU channel 0 interrupt is disabled '1': ICU channel 0 interrupt is enabled
[3:2]	EG1	ICU Edge Selection These bits provide edge selection for the input (ICU channel 1). '00': No edge detection (stop) '01': Rising edge detection '10': Falling edge detection '11': Both edges detection

Table 26-38. Input Capture Edge Register and Control Status Register (ICUn_ICEICS01) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	EG0	ICU Edge Selection These bits provide edge selection for the input (ICU channel 0). '00': No edge detection (stop) '01': Rising edge detection '10': Falling edge detection '11': Both edges detection

26.4.1.5 DMA Configuration Register (ICUn_DMACFG01)

The DMA Configuration Register (ICUn_DMACFG01) contain bits to enable/disable the DMA request for respective ICU channel.

DMA Configuration Register (ICUn_DMACFG01)

Figure 26-51. DMA Configuration Register (ICUn_DMACFG01)

ICUn_DMACFG01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	ENDMAREQ1	ENDMAREQ0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 26-39. DMA Configuration Register (ICUn_DMACFG01) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	ENDMAREQ1	Enable DMA Request This bit is used to enable/disable the DMA request (ICU channel 1). '0': ICU channel 1 DMA request is disabled '1': ICU channel 1 DMA request is enabled
[0]	ENDMAREQ0	Enable DMA Request This bit is used to enable/disable the DMA request (ICU channel 0). '0': ICU channel 0 DMA request is disabled '1': ICU channel 0 DMA request is enabled

26.4.1.6 Debug Register (ICUn_DEBUG01)

The Debug Register (ICUn_DEBUG01) configures debug support of the ICU channel.

Debug Register (ICUn_DEBUG01)

Figure 26-52. Debug Register (ICUn_DEBUG01)

ICUn_DEBUG01							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	DBGEN1	DBGEN0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 26-40. Debug Register (ICUn_DEBUG01) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	DBGEN1	Debug Enable This bit is used to enable/disable the debug mode support (ICU channel 1). '0': ICU channel 1 debug mode support is disabled '1': ICU channel 1 debug mode support is enabled When DBGEN1 is set to '1' and the processor is in debug state, edge detection operation is stopped for channel 1. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[0]	DBGEN0	Debug Enable This bit is used to enable/disable the debug mode support (ICU channel 0). '0': ICU channel 0 debug mode support is disabled '1': ICU channel 0 debug mode support is enabled When DBGEN0 is set to '1' and the processor is in debug state, edge detection operation is stopped for channel 0. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.

26.4.2 Operation of the 16-bit input capture

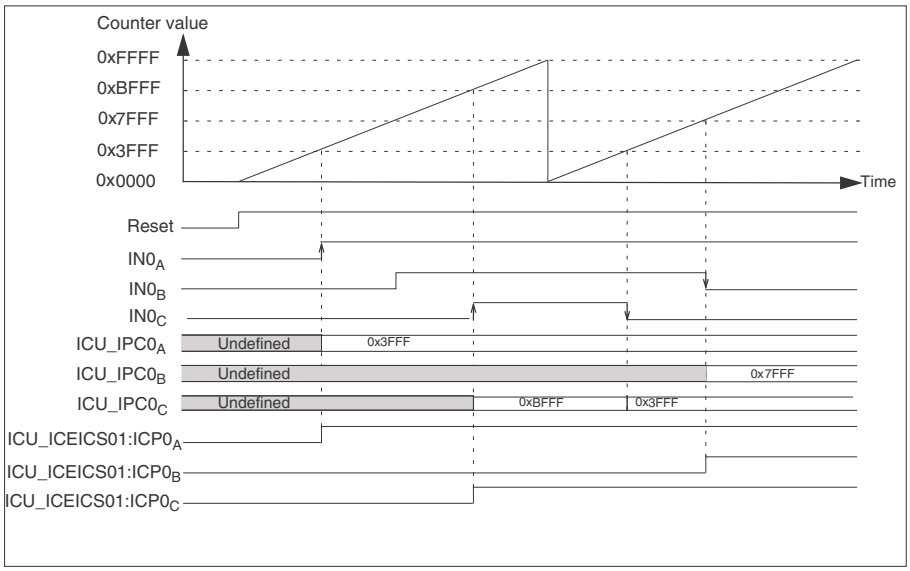
In 16-bit input capture operation, an interrupt can be generated upon detection of a specified edge, fetching the 16-bit Free Running Timer value and writing it to the Input Capture Data Register.

Example of input capture fetch timing

The following figure shows the FRT captured value (ICUn_IPC0) and capture interrupt (ICUn_ICEICS01:ICP) with respect to the different configurations for edge detection ICUn_ICEICS01:EG0[1:0].

- A: Rising edge
- B: Falling edge
- C: Both edges

Figure 26-53. Example of input capture fetch timing



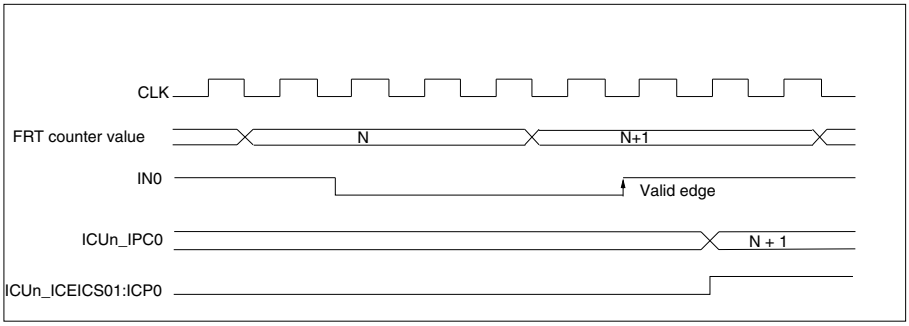
Notes:

1. The subscripts A, B and C show signals corresponding to a different edge configuration.
2. Only ICU channel 0 signals are shown in the figure.
3. For the minimum pulse width length of IN0 and IN1 refer to the device-specific datasheet.

Input capture input timing

The following figure shows the FRT captured value (ICUn_IPC0) and capture interrupt (ICUn_ICEICS01:ICP) timings with respect to input IN0.

Figure 26-54. Capture timing for input signals



26.4.3 Application of 16-bit I/O Timer

Period measurement and waveform observation activity can be performed using the 16-bit I/O Timer. This section describes the required 16-bit I/O Timer configurations and sequence.

Speed measurement

Measurement of time between two edges (period measurement)

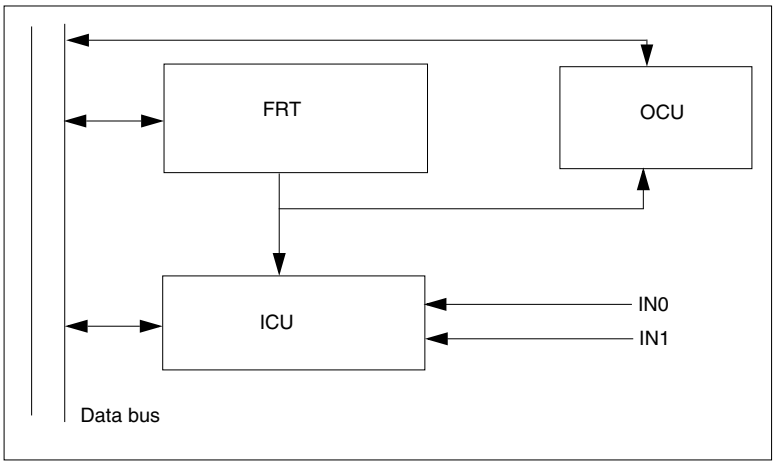
The register configuration required for period measurement is:

1. Set bits EG[0] and EG[1] of (ICUn_ICEICS01) to detect the active edge of IN0 or IN1.
2. Set bit IDSE0/IDSE1 of (ICUn_ICEICS01) to enable the output of the FRT counter clear signal to the selected FRT.
3. Set the bit ICUR of the corresponding FRTn:ETCCS to enable the FRT counter clear signal.
4. Configure resource mux configuration register to select the FRT.
5. After aquiring the active edge at the input of the ICU, it transfers the FRT value in Input Capture Data Register (ICUn_IPC01) of ICU.
   a. After getting the first edge at the input of the ICU it transfers the FRT value to the Input Capture Data Register (ICUn_IPC01) of the ICU, then resets the selected FRT.
   b. The second edge captures the FRT value in the Input Capture Data Register, then resets the FRT value.
   c. The CPU can read the captured time from the ICUn_IPC01 register which contains the time between the two edges.

Waveform observer

- The waveform observer activity is performed by the ICU, FRT and OCU
- The input signal INn needs a different waveform observer
- The operation of waveform observer is the same as speed measurement, with the OCU output compare register being programmed for maximum value
- The active edge should come before the FRT counter value reaches the value programmed in OCU or FRT overflow (if OCU is programmed for shorter time than FRT compare clear then the OCU compare match interrupt occurs instead of a FRT overflow interrupt)

Figure 26-55. Block diagram of the waveform observer



27. Programmable Pulse Generator

This chapter explains the function and operation of the Programmable Pulse Generator.

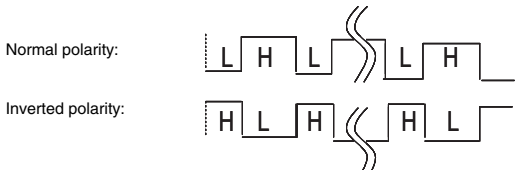
27.1 Outline of the Programmable Pulse Generator

Programmable Pulse Generators (PPGs) are used to generate one-shot (rectangular wave) output waveforms or pulse width modulation (PWM) output waveforms. With their software programmable cycle and duty capability, the ability to postpone the start of PWM output signal generation through software and trigger an AD conversion, the PPGs comfortably fit into a broad range of applications. To increase flexibility, PPGs can be configured as either one 16-bit resolution PWM channel or two independent 8-bit resolution PWM outputs. Moreover, the PPGs can operate in ramp mode, changing the output signal duty between defined start duty and end duty values.

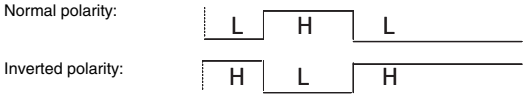
Features of the Programmable Pulse Generator

Programmable Pulse Generators (PPGs) are used to generate one-shot (rectangular wave) or Pulse Width Modulation (PWM) output waveforms. The features of this module are:

- Output waveforms: PPGs can generate the following kinds of waveforms
  - PWM waveform



  - One-shot waveform (rectangular wave)



- Clamped output during PPG off
  - In normal polarity: 'L' (Low) clamped output
  - In inverted polarity: 'H' (High) clamped output
- Count clock: This is selectable between the peripheral clock (CLKP = CLK_PERI0_PD2 for n ≤ 63, CLK_PERI1_PD2 for n ≥ 64) or a selected timer signal with an additional prescaler of 1, 1/4, 1/16 or 1/64
- Cycle setting range
  - Cycle setting range: Duty value ≤ cycle value ≤ 65535 (16-bit operation) or duty value ≤ cycle value ≤ 255 (8-bit operation)
  - Full range mode:

    Cycle = Count clock x PPGn_PCSR register value

    E.g: Count cycle = 32 MHz (31.25 ns), PPGn_PCSR value = 64000 (16-bit operation) Cycle = 31.25 ns x 64000 = 2 ms
  - All other modes:

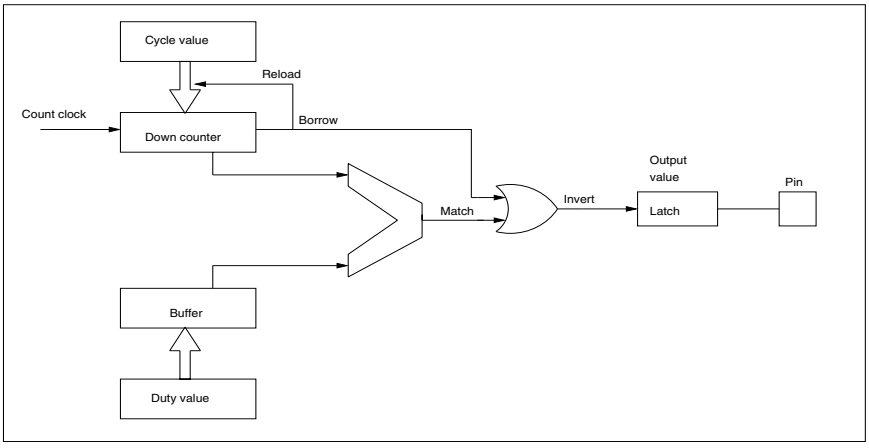
    Cycle = Count clock x (PPGn_PCSR register value + 1)

    E.g.: Count clock = 32 MHz (31.25 ns), PPGn_PCSR value = 63999 (16-bit operation) Period = 31.25 ns x (63999 + 1) = 2 ms

- Duty setting range
  - Duty setting range: 0 ≤ duty value ≤ cycle value (8-bit or 16-bit register value)
  - Full range mode:
    Duty = Count clock x PPGn_PDUT register value
  - All other modes:
    Duty = Count clock x (PPGn_PDUT register value + 1)
- Interrupt: There are six possibilities
  - Software trigger or external trigger (ETRG pin)
  - Counter borrow (cycle match)
  - Duty match
  - Counter borrow (cycle match) or duty match
  - Defined timing point match within the PPG cycle
  - End duty match during the ramp mode operation
- Activation trigger:
  - Software trigger
  - Internal trigger
  - External trigger (ETRG pins)
  - Common internal trigger, able to trigger all available PPG resources
- Special timing point within the PPG cycle can be configured such that when the PPG counter reaches the timing point value:
  - An Analog-to-Digital Conversion (ADC) trigger can be generated
  - An interrupt can be requested
  - A DMA request can be generated
- Ramp mode operation allows the PWM signal duty to sweep between configured values of the start duty and end duty while controlling the ramp slope over the selected timer period
- The PPG supports Microcontroller Unit (MCU) debug mode

Block diagram of the Programmable Pulse Generator

Figure 27-1. Block diagram of the Programmable Pulse Generator



Four PPGs form a group with common PPGn_GCN registers.

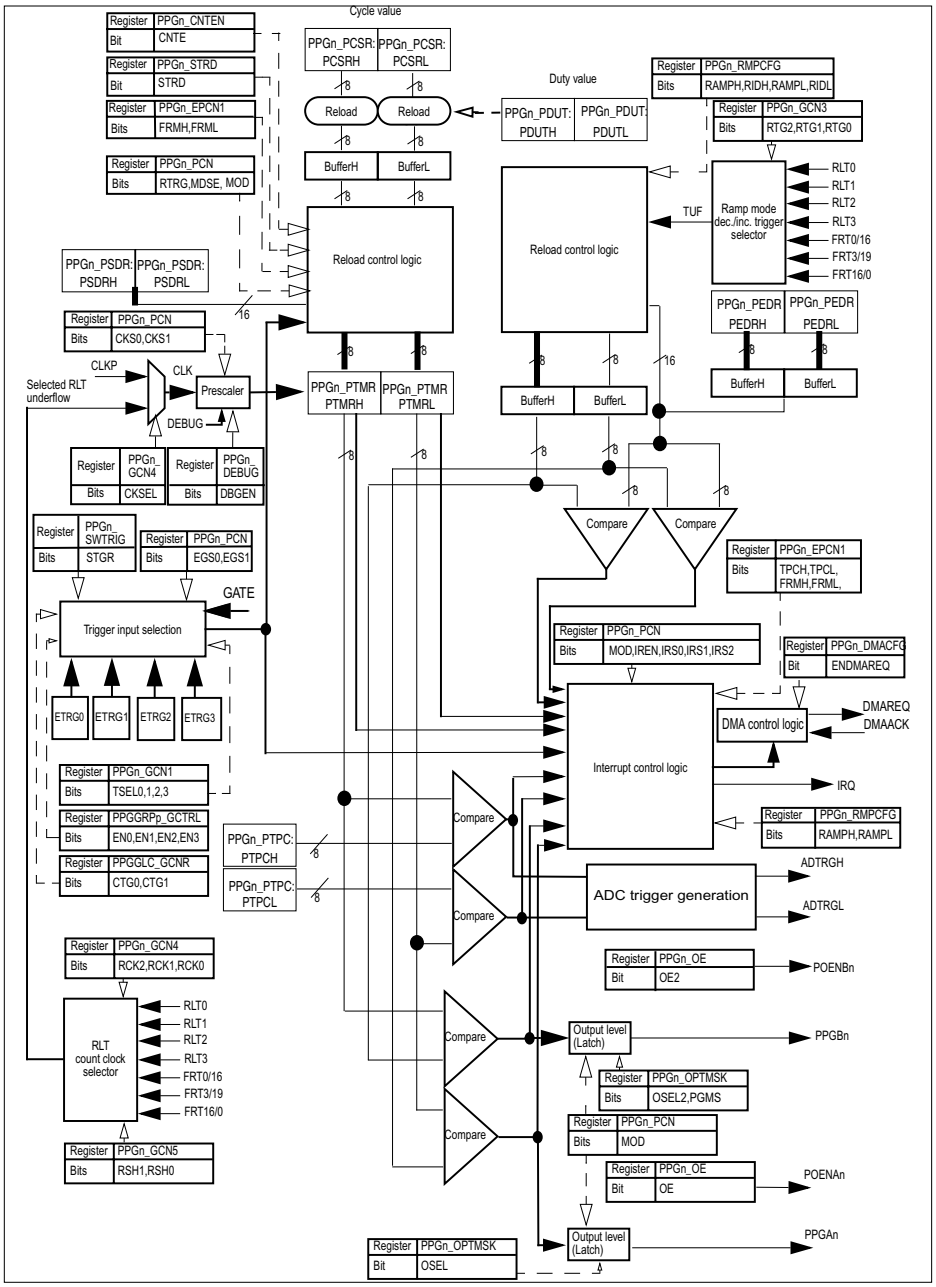
In Figure 27-2 the following notation is used:

n = Number of the PPG

p = INT(n/4) group number of the PPG

g = INT(p/16) number of the peripheral group

Figure 27-2. Detailed block diagram of the Programmable Pulse Generator



Notes:

1. To configure Reload Timer (RLT) signals, see 28. 32-Bit Reload Timer. RLT can be used as a normal RLT if it is not required for PPG operation.
2. To configure Free running Timer (FRT) signals, see 26. 16-Bit I/O Timer. FRT can be used as a normal FRT if it is not required for PPG operation.
3. PPGAn/PPGBn can be gated before being output to device pins PPGn_PPGA/PPGn_PPGB. For more details, refer to 14. Resource Input Configuration.
4. For the minimum pulse width length of ETRG refer to device specific datasheet.

5. CLKP is the relevant peripheral clock.
   For n ≤ 63, CLKP = CLK_PERI0_PD2.
   For n ≥ 64, CLKP = CLK_PERI1_PD2.

27.2 Programmable Pulse Generator registers

This section describes the registers of the Programmable Pulse Generator in detail.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the PPG channel.

The suffix ’p’ in the register name indicates that the register is an instance ’p’ of the PPG group.

The suffix ’g’ in the register name indicates that the register is in peripheral group ’g’.

Registers of Programmable Pulse Generator

The following PPG registers are available:

- PPG Control Status Register (PPGn_PCN)
- Interrupt Flag Clear Register (PPGn_IRQCLR)
- Software Trigger Activation Register (PPGn_SWTRIG)
- Output Enable Register (PPGn_OE)
- Timer Enable Operation Register (PPGn_CNTEN)
- Output Mask and Polarity Selection Register (PPGn_OPTMSK)
- Ramp Configuration Register (PPGn_RMPCFG)
- Start Delay Mode Register (PPGn_STRD)
- PPG Trigger Clear Flag Register (PPGn_TRIGCLR)
- Extended PPG Control Status Register 1 (PPGn_EPCN1)
- Extended PPG Control Status Register 2 (PPGn_EPCN2)
- General Control Register 1 (PPGn_GCN1)
- General Control Register 3 (PPGn_GCN3)
- General Control Register 4 (PPGn_GCN4)
- General Control Register 5 (PPGn_GCN5)
- PPG Cycle Setting Register (PPGn_PCSR)
- PPG Duty Setting Register (PPGn_PDUT)
- PPG Timer Register (PPGn_PTMR)
- PPG Start Delay Register (PPGn_PSDR)
- PPG Timing Point Capture Register (PPGn_PTPC)
- PPG End Duty Register (PPGn_PEDR)
- PPG DMA Configuration Register (PPGn_DMACFG)
- PPG Debug Enable Register (PPGn_DEBUG)

The following register is available for a group of Programmable Pulse Generators:

- Group Control Register (PPGGRPp_GCTRL)

The following register is available for the Programmable Pulse Generators in a peripheral group:

- PPG Global Control Register (PPGGLCg_GCNR)

Memory layout of Programmable Pulse Generator registers

Table 27-1. Memory layout of Programmable Pulse Generator registers

Offset	+1	+0
0x00000000	PPGn_PCN 00000000 00000000	
0x00000002	PPGn_SWTRIG 00000000	PPGn_IRQCLR 00000000
0x00000004	PPGn_CNTEN 00000000	PPGn_OE 00000000
0x00000006	PPGn_RMPCFG 00000000	PPGn_OPTMSK 00000000
0x00000008	PPGn_TRIGCLR 00000000	PPGn_STRD 00000000
0x0000000A	PPGn_EPCN1 00000000 00000000	
0x0000000C	PPGn_EPCN2 00000000 00000000	
0x0000000E	PPGn_GCN3 00000000	PPGn_GCN1 00000000
0x00000010	PPGn_GCN5 00000000	PPGn_GCN4 00000110
0x00000012	PPGn_PCSR XXXXXXXX XXXXXXXX	
0x00000014	PPGn_PDUT XXXXXXXX XXXXXXXX	
0x00000016	PPGn_PTMR 11111111 11111111	
0x00000018	PPGn_PSDR 00000000 00000000	
0x0000001A	PPGn_PTPC 00000000 00000000	
0x0000001C	PPGn_PEDR 00000000 00000000	
0x0000001E	PPGn_DEBUG 00000000	PPGn_DMACFG 00000000

Table 27-2. Memory layout of PPG Group Control Register

Offset	+1	+0
0x00000000	reserved XXXXXXXX	PPGGRPp_GCTRL 00000000

Table 27-3. Memory layout of PPG Global Control Register

Offset	+1	+0
0x00000000	reserved XXXXXXXX	PPGGLCg_GCNR 00000000

27.2.1 PPG Control Status Register (PPGn_PCN)

The PPG Control Status Register (PPGn_PCN) controls the operations and reflects the status of the PPG.

PPG Control Status Register (PPGn_PCN)

Figure 27-3. PPG Control Status Register (PPGn_PCN)

PPGn_PCN															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	MDSE	RTRG	CKS[1]	CKS[0]	read0	MOD	EGS[1]	EGS[0]	IREN	IRQF	IRS[2]	IRS[1]	IRS[0]	read0
Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	Rp	RpWp	RpWp	RpWp	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-4. PPG Control Status Register (PPGn_PCN) bits

Bit Position	Bit Field Name	Bit Description
[15:14]	read0	-
[13]	MDSE	Mode Selection '0': PWM operation '1': One-shot operation When the mode selection bit is set to '0', a PWM operation is enabled to generate pulses in sequence. When the mode selection bit is set to '1', the pulse output takes place only once.
[12]	RTRG	Restart Enable '0': Disable restart '1': Enable restart When the restart enable bit is set to '1', a trigger (software/internal/ external) restarts the PPG operation (depending on the configuration of the triggers).
[11:10]	CKS	Counter Clock Selection CKS[1:0] down counter count clock selection. '00': Clock selected by PPGn_GCN4:CKSEL '01': Clock selected by PPGn_GCN4:CKSEL divided by 4 '10': Clock selected by PPGn_GCN4:CKSEL divided by 16 '11': Clock selected by PPGn_GCN4:CKSEL divided by 64
[9]	read0	-

Table 27-4. PPG Control Status Register (PPGn_PCN) bits

Bit Position	Bit Field Name	Bit Description
[8]	MOD	PPG 16-bit/8-bit Operation Mode '0': 16-bit mode '1': 8-bit mode When the MOD bit is set to '0', a PWM output signal is defined with 16-bit resolution. When the MOD bit is set to '1', a PWM output signal is defined with 8-bit resolution.
[7:6]	EGS	Trigger Input Edge Selection '00': No edge selected; triggering of PPG only possible by PPGn_SWTRIG:STRG '01': Rising edge '10': Falling edge '11': Both edges (rising and falling)
[5]	IREN	Interrupt Request Enable '0': Disable interrupt requests '1': Enable interrupt requests
[4]	IRQF	Interrupt Request Flag '0': No interrupt request '1': Interrupt request This flag is set to '1' when the interrupt cause condition occurs; it depends on the setting of the PPGn_PCN:IRS[2:0] bits. PPGn_PCN:IRS defines the interrupt causes.
[3:1]	IRS	Interrupt Cause Selection These bits select the operation in which to generate an interrupt request. '000': Software trigger or external trigger input '001': Counter borrow (16-bit, or upper (i.e. higher) and lower 8-bits if PPGn_PCN:MOD = '1') '010': The counter matches the duty value (16-bit, or upper and lower 8-bits if PPGn_PCN:MOD = '1') '011': Counter borrow or the counter equals the duty value (16-bit, or upper and lower 8-bits if PPGn_PCN:MOD = '1') '100': Timing point capture (16-bit, or upper and lower 8-bits if PPGn_PCN:MOD = '1') '101': End duty value match (16-bit, or upper and lower 8-bits if PPGn_PCN:MOD = '1') Others: Reserved
[0]	read0	-

27.2.2 Interrupt Flag Clear Register (PPGn_IRQCLR)

The Interrupt Flag Clear Register (PPGn_IRQCLR) clears the interrupt request flag PPGn_PCN:IRQF.

Interrupt Flag Clear Register (PPGn_IRQCLR)

Figure 27-4. Interrupt Flag Clear Register (PPGn_IRQCLR)

PPGn_IRQCLR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	IRQCLR
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1
0	0	0	0	0	0	0	0

Table 27-5. Interrupt Flag Clear Register (PPGn_IRQCLR) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	IRQCLR	Interrupt Request Flag Clear bit '0': No effect '1': Clears the following flags: PPGn_PCN:IRQF flag; PPGn_EPCN2:TCH flag; PPGn_EPCN2:TCL flag; PPGn_EPCN2:EDMH flag; PPGn_EPCN2:EDML flag; PPGn_EPCN2:DTH flag; PPGn_EPCN2:DTL flag; PPGn_EPCN2:PRDH flag; PPGn_EPCN2:PRDL flag; and PPGn_EPCN1:TRIG flag Reading this bit always returns '0'.

27.2.3 Software Trigger Activation Register (PPGn_SWTRIG)

The Software Trigger Activation Register (PPGn_SWTRIG) controls the software triggered activation of the PPG.

Software Trigger Activation Register (PPGn_SWTRIG)

Figure 27-5. Software Trigger Activation Register (PPGn_SWTRIG)

PPGn_SWTRIG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	STGR
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1
0	0	0	0	0	0	0	0

Table 27-6. Software Trigger Activation Register (PPGn_SWTRIG) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	STGR	Software Trigger '0': The operation is unaffected by writing '0' (the read value is always '0') '1': Software trigger activation When the software trigger bit is set to '1', a software trigger is generated to activate the PPG separately from the generation of an internal or external trigger (PPGGRPp_GCTRL:EN bit, Reload Timer output, ETRG input). This trigger is independent of the setting of the edge selection bits PPGn_PCN:EGS1 and PPGn_PCN:EGS0.

27.2.4 Output Enable Register (PPGn_OE)

The Output Enable Register (PPGn_OE) controls the PPG output.

Output Enable Register (PPGn_OE)

Figure 27-6. Output Enable Register (PPGn_OE)

PPGn_OE							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	OE2	OE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-7. Output Enable Register (PPGn_OE) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	OE2	PPGB Output Enable '0': Output disabled '1': Output enabled In 8-bit operation mode (i.e. PPGn_PCN:MOD = '1') this bit enables the outputting of the PPGB signal (i.e. an 8-bit resolution PWM signal configured by the upper 8 register bits). In 16-bit operation mode (PPGn_PCN:MOD = '0') the OE2 control bit can be set to '1' to generate the 16-bit PPG signal also on PPGB output pin. Additionally if the PPGn_OPTMSK:OSEL2 bit is set to '1', the 16-bit PPG signal with inverted polarity is sent to the PPGB output pin.
[0]	OE	PPGA Output Enable '0': Output disabled '1': Output enabled In 8-bit operation mode (PPGn_PCN:MOD = '1') this bit enables the outputting of the PPGA signal (i.e. the 8-bit resolution PWM signal configured by the lower 8 register bits). In 16-bit operation mode (PPGn_PCN:MOD = '0') the OE control bit is used to enable the output of the 16-bit PWM signal on the PPGA pin.

27.2.5 Timer Enable Operation Register (PPGn_CNTEN)

The Timer Enable Operation Register (PPGn_CNTEN) controls the PPG operation.

Timer Enable Operation Register (PPGn_CNTEN)

Figure 27-7. Timer Enable Operation Register (PPGn_CNTEN)

PPGn_CNTEN							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	CNTE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 27-8. Timer Enable Operation Register (PPGn_CNTEN) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	CNTE	Count Enable This bit enables the operation of the PPG. '0': Stop '1': Operation

27.2.6 Output Mask and Polarity Selection Register (PPGn_OPTMSK)

The Output Mask and Polarity Selection Register (PPGn_OPTMSK) controls the PPG output polarity selection.

Output Mask and Polarity Selection Register (PPGn_OPTMSK)

Figure 27-8. Output Mask and Polarity Selection Register (PPGn_OPTMSK)

PPGn_OPTMSK							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	PGMS	OSEL2	OSEL
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-9. Output Mask and Polarity Selection Register (PPGn_OPTMSK) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	PGMS	PPG Output Mask Selection '0': No output mask '1': Output mask When the PPG output mask selection bit is set to '1', the PPG output can be clamped at L or H regardless of the mode, cycle and duty settings. The output level can be specified using the output polarity specification bit (PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2). If PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2 = '0', then the output level on PPGA/PPGB is 'L' If PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2 = '1', then the output level on PPGA/PPGB is 'H'
[1]	OSEL2	PPGB Output Polarity Specification (Upper 8-bits) '0': PPGB output signal has normal polarity '1': PPGB output signal has inverted polarity In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') if the OE2 control bit is set to '1', the 16-bit PPG signal is also generated on the PPGB output pin. In addition, if the OSEL2 bit is set to '1', the 16-bit PPG signal with inverted polarity is sent to the PPGB output pin.
[0]	OSEL	PPGA Output Polarity Specification '0': PPGA output signal has normal polarity '1': PPGA output signal has inverted polarity

27.2.7 Ramp Configuration Register (PPGn_RMPCFG)

The Ramp Configuration Register (PPGn_RMPCFG) controls the ramp mode operation of the PPG.

Ramp Configuration Register (PPGn_RMPCFG)

Figure 27-9. Ramp Configuration Register (PPGn_RMPCFG)

PPGn_RMPCFG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	RIDH	RIDL	RAMPH	RAMPL
Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-10. Ramp Configuration Register (PPGn_RMPCFG) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	RIDH	Duty Increment/Decrement in Ramp Mode (upper 8-bits) '0': Increment the PWM duty of the PPGB output in ramp mode until the end duty value is reached '1': Decrement the PWM duty of the PPGB output in ramp mode until the end duty value is reached
[2]	RIDL	Duty Increment/Decrement in Ramp Mode (16-bit or lower 8-bits). '0': Increment the PWM duty of the PPGA output in ramp mode until the end duty value is reached '1': Decrement the PWM duty of the PPGA output in ramp mode until the end duty value is reached In 16-bit operation mode (PPGn_PCN:MOD = '0') the RIDL bit controls the duty increment/decrement function of the 16-bit PWM signal. During 8-bit operation (PPGn_PCN:MOD = '1') it is dedicated to the lower 8-bit PWM output.

Table 27-10. Ramp Configuration Register (PPGn_RMPCFG) bits

Bit Position	Bit Field Name	Bit Description
[1]	RAMPH	Ramp Mode Selection (upper (i.e. higher) 8-bits) This bit enables ramp mode PWM operation for the upper 8-bits if PPGn_PCN:MOD = '1'. In this mode PWM duty is incremented (PPGn_RMPCFG:RIDH is '0') or decremented (PPGn_RMPCFG:RIDH is '1') with every external trigger signal (one of seven possible timer signals). '0': Disable ramp mode for PPGB output '1': Enable ramp mode for PPGB output The PWM output waveform starts with the duty defined by the PPGn_PDUT:PDUTH register and the duty value is incremented/decremented until the last duty value is reached, which is defined by the register PPGn_PEDR:PEDRH. The actual update of the PWM output duty takes place only at the end of the PWM cycle.
[0]	RAMPL	Ramp Mode Selection (16-bit or low 8-bit part) This bit enables ramp mode PWM operation for the lower 8-bits if PPGn_PCN:MOD = '1'. During 16-bit operation, the bit controls the ramp mode of 16-bit PWM. In this mode the PWM duty is incremented (PPGn_RMPCFG:RIDL is '0') or decremented (PPGn_RMPCFG:RIDL is '1') with every external trigger signal (selected one of seven possible timer signals). '0': Disable ramp mode for PPGA output '1': Enable ramp mode for PPGA output The PWM output waveform starts with the duty defined by PPGn_PDUT/PPGn_PDUT:PDUTL register and the duty value is incremented/decremented until the end duty value is reached defined by the register PPGn_PEDR/PPGn_PEDR:PEDRL. Actual update of the PWM output duty takes place only at the end of the PWM cycle.

27.2.8 Start Delay Mode Register (PPGn_STRD)

The Start Delay Mode Register (PPGn_STDRn) controls the start delay of PWM generation.

Start Delay Mode Register (PPGn_STRD)

Figure 27-10. Start Delay Mode Register (PPGn_STRD)

PPGn_STRD							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	STRD
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 27-11. Start Delay Mode Register (PPGn_STRD) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	STRD	Start Delay Mode '0': Delayed start of PWM output generation is disabled '1': Delayed start of PWM output generation is enabled When the STRD bit is set to '1', the PWM output generation is delayed by PSDR + 1 cycles of the PPG counter clock. If the full range mode is activated (PPGn_EPCN1:FRMH/FRML = '1'), the start is delayed by PSDR cycles of the PPG counter clock.

27.2.9 PPG Trigger Clear Flag Register (PPGn_TRIGCLR)

The PPG Trigger Clear Flag Register (PPGn_TRIGCLR) clears the PPG start/trigger event flag PPGn_EPCN1:TRIG.

PPG Trigger Clear Flag Register (PPGn_TRIGCLR)

Figure 27-11. PPG Trigger Clear Flag Register (PPGn_TRIGCLR)

PPGn_TRIGCLR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	TRGCLR
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1
0	0	0	0	0	0	0	0

Table 27-12. PPG Trigger Clear Flag Register (PPGn_TRIGCLR)

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	TRGCLR	PPG Start/Trigger Event Flag Clear bit '0': No effect '1': Clears the PPGn_EPCN1:TRIG flag Reading this bit always returns '0'.

27.2.10 Extended PPG Control Status Register 1 (PPGn_EPCN1)

The Extended PPG Control Status Register 1 (PPGn_EPCN1) controls the operation and status of the PPG.

Extended PPG Control Status Register 1 (PPGn_EPCN1)

Figure 27-12. Extended PPG Control Status Register 1 (PPGn_EPCN1)

PPGn_EPCN1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	TRIG	read0	read0	FRMH	FRML	read0	read0	read0	read0	read0	TPCH	TPCL	read0
Rp0	Rp0	Rp0	Rp	Rp0	Rp0	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-13. Extended PPG Control Status Register 1 (PPGn_EPCN1) bits

Bit Position	Bit Field Name	Bit Description
[15:13]	read0	-
[12]	TRIG	PPG Start/Trigger Event Flag This is a read-only bit. '0': No interrupt '1': Interrupt The TRIG flag is set to '1' when the PWM output generation is started. In start delay mode (i.e. PPGn_STRD:STRD = '1') this happens at the end of a defined start delay. For all other operation modes (i.e. PPGn_STRD:STRD = '0'), the flag becomes '1' when a PPG trigger event is detected. In 8-bit mode with PPGn_STRD:STRD = '1', the PPGn_EPCN1:TRIG flag is set from both 8-bit individual channels. This bit is cleared by writing '1' to either PPGn_TRIGCLR:TRGCLR or PPGn_IRQCLR:IRQCLR.
[11:10]	read0	-

Table 27-13. Extended PPG Control Status Register 1 (PPGn_EPCN1) bits

Bit Position	Bit Field Name	Bit Description
[9]	FRMH	Full Range Mode (Upper 8-bits) '0': The PPGB output signal has a cycle of PPGn_PCSR + 1 and duty of PPGn_PDUT + 1 count clock cycles '1': The PPGB output signal has a cycle of PPGn_PCSR and duty of PPGn_PDUT count clock cycles When this bit is set to '1', the limit case PPGn_PDUT = PPGn_PCSR generates level 'H' on the PPGB output and for PPGn_PDUT = '0', level 'L' is generated all time.
[8]	FRML	Full Range Mode (16-bit or lower 8-bits) '0': The PPGA output signal has a cycle of PPGn_PCSR + 1 and duty of PPGn_PDUT + 1 count clock cycles '1': The PPGA output signal has a cycle of PPGn_PCSR and duty of PPGn_PDUT count clock cycles When this bit is set to '1', the limit case PPGn_PDUT = PPGn_PCSR generates level 'H' on the PPGA output, and for PDUT = '0', level 'L' is generated all the time.
[7:3]	read0	-
[2]	TPCH	Timing Point Capture Selection (upper 8-bits). '0': Disable timing point capture mode for PPGB output '1': Enable timing point capture mode for PPGB output If this bit is set to '1' and PPGn_PCN:MOD = '1', the matching of the PPG counter (upper 8-bits) and PPGn_PTPC:PTPCH generates an ADC trigger signal.
[1]	TPCL	Timing Point Capture Selection (16-bit or lower 8-bits). '0': Disable timing point capture mode for the PPGA output '1': Enable timing point capture mode for the PPGA output If this bit is set to '1' and PPGn_PCN:MOD = '1', the matching of the PPG counter (lower 8-bits) and PPGn_PTPC:PTPCL bit description generates an ADC trigger signal. Accordingly, during 16-bit operation, the ADC trigger signal is generated if the 16-bit PPG counter value matches the PPGn_PTPC.
[0]	read0	-

27.2.11 Extended PPG Control Status Register 2 (PPGn_EPCN2)

The Extended PPG Control Status Register 2 (PPGn_EPCN2) controls the operation and status of the PPG.

Extended PPG Control Status Register 2 (PPGn_EPCN2)

Figure 27-13. Extended PPG Control Status Register 2 (PPGn_EPCN2)

PPGn_EPCN2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TCHCLR	TCLCLR	EDMHCLR	EDMLCLR	DTHCLR	DTLCLR	PRDHCLR	PRDLCLR	TCH	TCL	EDMH	EDML	DTH	DTL	PRDH	PRDL
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-14. Extended PPG Control Status Register 2 (PPGn_EPCN2) bits

Bit Position	Bit Field Name	Bit Description
[15]	TCHCLR	Timing Point Capture Flag (Upper 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:TCH flag Reading this bit always returns '0'.
[14]	TCLCLR	Timing Point Capture Flag (16-bit or lower 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:TCL flag Reading this bit always returns '0'.
[13]	EDMHCLR	End Duty Match Flag in Ramp Mode (Upper 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:EDMH flag Reading this bit always returns '0'.
[12]	EDMLCLR	End Duty Match Flag in Ramp Mode (16-bit or lower 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:EDML flag Reading this bit always returns '0'.
[11]	DTHCLR	Duty Match Flag (upper 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:DTH flag Reading this bit always returns '0'.

Table 27-14. Extended PPG Control Status Register 2 (PPGn_EPCN2) bits

Bit Position	Bit Field Name	Bit Description
[10]	DTLCLR	Duty Match Flag (16-bit or lower 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:DTL flag Reading this bit always returns '0'.
[9]	PRDHCLR	Cycle Match Flag (upper 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:PRDH flag Reading this bit always returns '0'.
[8]	PRDLCLR	Cycle Match Flag (16-bit or lower 8-bits) Clear bit '0': No effect '1': Clears the PPGn_EPCN2:PRDL flag Reading this bit always returns '0'.
[7]	TCH	Timing Point Capture Flag (Upper 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request The TCH flag is set to '1' when the PPG counter value PTMRH reaches the timing point defined by the PPGn_PTPC:PTPCH register. In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the TCH flag has no meaning. This bit is cleared by writing '1' to either PPGn_EPCN2:TCHCLR or PPGn_IRQCLR:IRQCLR.
[6]	TCL	Timing Point Capture Flag (16-bit or lower 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the TCL flag is set by matching the 16-bit registers PPGn_PTMR and PPGn_PTPC. If PPGn_PCN:MOD = '1', the flag is a result of a comparison made between PPGn_PTMR:PTMRL and PPGn_PTPC:PTPCL. This bit is cleared by writing '1' to either PPGn_EPCN2:TCLCLR or PPGn_IRQCLR:IRQCLR.
[5]	EDMH	End Duty Match Flag in Ramp Mode (upper 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request The EDMH flag is set to '1' when the current PWM duty has reached the duty value defined by the PPGn_PEDR:PEDRH register in ramp operation mode. If the control bit PPGn_RMPCFG:RAMPH is '0' or in 16-bit operation mode (i.e. PPGn_PCN:MOD = '1') the EDMH flag has no meaning. This bit is cleared by writing '1' to either PPGn_EPCN2:EDMHCLR or PPGn_IRQCLR:IRQCLR.

Table 27-14. Extended PPG Control Status Register 2 (PPGn_EPCN2) bits

Bit Position	Bit Field Name	Bit Description
[4]	EDML	End Duty Match Flag in Ramp Mode (16-bit or lower 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the EDML flag is set by matching the current duty value and PPGn_PEDR. If PPGn_PCN:MOD = '1', the flag results when the duty value defined by PPGn_PEDR:PEDRL is reached. This bit is cleared by writing '1' to either PPGn_EPCN2:EDMLCLR or PPGn_IRQCLR:IRQCLR.
[3]	DTH	Duty Match Flag (Upper 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request The DTH flag is set to '1' by matching registers PPGn_PTMR:PTMRH and PPGn_PDUT:PDUTH. In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the DTH flag has no meaning. This bit is cleared by writing '1' to either PPGn_EPCN2:DTHCLR or PPGn_IRQCLR:IRQCLR.
[2]	DTL	Duty Match Flag (16-bit or lower 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the DTL flag is set by matching the 16-bit registers PPGn_PTMR and PPGn_PDUT. If PPGn_PCN:MOD = '1', the flag is a result of comparing PPGn_PTMR:PTMRL and PPGn_PDUT:PDUTL. This bit is cleared by writing '1' to either PPGn_EPCN2:DTLCLR or PPGn_IRQCLR:IRQCLR.
[1]	PRDH	Cycle Match Flag (Upper 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request The PRDH flag is set to '1' in case of a PPGn_PTMR:PTMRH counter underflow. In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the PRDH flag has no meaning. This bit is cleared by writing '1' to either PPGn_EPCN2:PRDHCLR or PPGn_IRQCLR:IRQCLR.
[0]	PRDL	Cycle Match Flag (16-bit or lower 8-bits) This is a read-only bit. '0': No interrupt request '1': Interrupt request In 16-bit operation mode (i.e. PPGn_PCN:MOD = '0') the PRDL flag is set by the 16-bit PPGn_PTMR counter underflow. If PPGn_PCN:MOD = '1', the flag results because of a PPGn_PTMR:PTMRL underflow. This bit is cleared by writing '1' to either PPGn_EPCN2:PRDLCLR or PPGn_IRQCLR:IRQCLR.

27.2.12 General Control Register 1 (PPGn_GCN1)

The General Control Register 1 (PPGn_GCN1) selects a trigger input to the PPG.

General Control Register 1 (PPGn_GCN1)

Figure 27-14. General Control Register 1 (PPGn_GCN1)

PPGn_GCN1							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	TSEL[3]	TSEL[2]	TSEL[1]	TSEL[0]
Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-15. General Control Register 1 (PPGn_GCN1) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-

Table 27-15. General Control Register 1 (PPGn_GCN1) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	TSEL	Trigger Select This field selects a trigger input to a PPG. TSEL[3:0] activation trigger specification: '0000': EN0 bit (PPGGRPp_GCTRL register) '0001': EN1 bit (PPGGRPp_GCTRL register) '0010': EN2 bit (PPGGRPp_GCTRL register) '0011': EN3 bit (PPGGRPp_GCTRL register) '0100': 32-bit Reload Timer 0 output '0101': 32-bit Reload Timer output selected by RICFGRg_PPGGRPpRLTTRG1 register '0110': CTG0 bit (PPGGCLg_GCNR register) '0111': CTG1 bit (PPGGCLg_GCNR register) '1000': External trigger selected by the RICFGRg_PPGGRPpETRG0 register '1001': External trigger selected by the RICFGRg_PPGGRPpETRG1 register '1010': External trigger selected by the RICFGRg_PPGGRPpETRG2 register '1011': External trigger selected by the RICFGRg_PPGGRPpETRG3 register Others: Disabled PPG0 to PPG3 as selected are activated when the edge specified by the Trigger Input Edge Selection bits (PPGn_PCN:EGS[1:0]) are detected by the specified activation trigger.

Note: The initial value of PPGn_GCN1 for the PPG channel within a group of PPGs is: PPG0 = 0x00, PPG1 = 0x01, PPG2 = 0x02, PPG3 = 0x03

27.2.13 General Control Register 3 (PPGn_GCN3)

The General Control Register 3 (PPGn_GCN3) selects a trigger input for the ramp mode duty increment/decrement operation to a PPG. One of seven input signals can be selected.

General Control Register 3 (PPGn_GCN3)

Figure 27-15. General Control Register 3 (PPGn_GCN3)

PPGn_GCN3							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	RTG[2]	RTG[1]	RTG[0]
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-16. General Control Register 3 (PPGn_GCN3) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2:0]	RTG	Ramp Mode Trigger Select RTG[2:0] trigger specification: '000': Reload Timer 0 underflow '001': Reload Timer 1 underflow '010': Reload Timer 2 underflow '011': Reload Timer 3 underflow '100': FRT0 compare clear match (n = 0..63) FRT16 compare clear match (n = 64..127) '101': FRT3 compare clear match (n = 0..63) FRT19 compare clear match (n = 64..127) '110': FRT16 compare clear match (n = 0..63) FRT0 compare clear match (n = 64..127) Others: FRT16 compare clear match (n = 0..63) FRT0 compare clear match (n = 64..127)

27.2.14 General Control Register 4 (PPGn_GCN4)

The General Control Register 4 (PPGn_GCN4) selects an input signal from a timer that can be used as the PPG count clock. One of seven input signals can be selected. The chosen input signal becomes the PPG count clock (prescaler input) only if the PPGn_GCN4:CKSEL bit is set to 1.

General Control Register 4 (PPGn_GCN4)

Figure 27-16. General Control Register 4 (PPGn_GCN4)

PPGn_GCN4							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	CKSEL	RCK[2]	RCK[1]	RCK[0]
Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	1	1	0

Table 27-17. General Control Register 4 (PPGn_GCN4) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	CKSEL	Prescaler Input Selection CKSEL '0': Use CLKP as the prescaler input '1': Use the input signal selected by PPGn_GCN4:RCK[2:0] with a delay selected by PPGn_GCN5:RSH[1:0] as the prescaler input

Table 27-17. General Control Register 4 (PPGn_GCN4) bits

Bit Position	Bit Field Name	Bit Description
[2:0]	RCK	Timer Count Clock Select This field selects an input signal that can be used as PPG count clock. One of seven input signals can be selected. The chosen input signal ‚Äì delayed as selected by PPGn_GCN5:RSH[1:0] ‚Äì becomes the PPG count clock (prescaler input) only if bit PPGn_GCN4:CKSEL is set to '1'. RCK[2:0] input signal selected: '000': Reload Timer 0 underflow '001': Reload Timer 1 underflow '010': Reload Timer 2 underflow '011': Reload Timer 3 underflow '100': FRT0 compare clear match (n = 0..63) FRT16 compare clear match (n = 64..127) '101': FRT3 compare clear match (n = 0..63) FRT19 compare clear match (n = 64..127) '110': FRT16 compare clear match (n = 0..63) FRT0 compare clear match (n = 64..127) Others: FRT16 compare clear match (n = 0..63) FRT0 compare clear match (n = 64..127)

27.2.15 General Control Register 5 (PPGn_GCN5)

The General Control Register 5 (PPGn_GCN5) introduces a fine delay (phase shift) of an input signal that can be used as the PPG count clock. The phase shift can be defined separately for every PPG module. If this feature is used, the selected timer should be set to a period of at least 13 peripheral clock cycles (13 CLKP).

General Control Register 5 (PPGn_GCN5)

Figure 27-17. General Control Register 5 (PPGn_GCN5)

PPGn_GCN5							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	RSH[1]	RSH[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-18. General Control Register 5 (PPGn_GCN5) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1:0]	RSH	Phase Shift This field introduces a fine delay (phase shift) to the input signal selected by PPGn_GCN4:RCK[2:0] that can be used as the PPG count clock. The phase shift can be defined separately for every PPG module. If this feature is used, the selected timer should be set to a period of at least 13 peripheral clock cycles (13 CLKP). RSH[1:0] fine delay '00': No delay '01': 4 CLKP cycles delay '10': 8 CLKP cycles delay '11': 12 CLKP cycles delay

27.2.16 PPG Cycle Setting Register (PPGn_PCSR)

The PPG Cycle Setting Register (PPGn_PCSR) controls the cycle of the PPG. In an 8-bit operation mode PPGn_PCSR:PCSRH defines the cycle of the PPGB output, and PPGn_PCSR:PCSRL sets the cycle of the PPGA signal.

PPG Cycle Setting Register (PPGn_PCSR)

Figure 27-18. PPG Cycle Setting Register (PPGn_PCSR)

PPGn_PCSR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PCSRH[7]	PCSRH[6]	PCSRH[5]	PCSRH[4]	PCSRH[3]	PCSRH[2]	PCSRH[1]	PCSRH[0]	PCSRL[7]	PCSRL[6]	PCSRL[5]	PCSRL[4]	PCSRL[3]	PCSRL[2]	PCSRL[1]	PCSRL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 27-19. PPG Cycle Setting Register (PPGn_PCSR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	PCSRH	PPG Cycle Setting Register The PPG Cycle Setting Register (PCSRn) controls the cycle of the PPG. In 8-bit operation mode PCSRH defines the cycle of the PPGB output. The PPG Cycle Setting Registers come with buffers. Transfers from the buffers to the counter take place automatically upon counter borrow. After the PPG Cycle Setting Registers have been written, be sure to set PPG Duty Setting Registers PPGn_PDUT. Reading this register returns the buffered value, i.e. actual cycle setting.
[7:0]	PCSRL	PPG Cycle Setting Register The PPG Cycle Setting Register (PCSRn) controls the cycle of the PPG. In 8-bit operation mode PCSRL defines the cycle of the PPGA output. The PPG Cycle Setting Registers come with buffers. Transfers from the buffers to the counter take place automatically upon counter borrow. After the PPG Cycle Setting Registers have been written to, set the PPG Duty Setting Register PPGn_PDUT. Reading this register returns the buffered value, i.e. actual cycle setting.

27.2.17 PPG Duty Setting Register (PPGn_PDUT)

The PPG Duty Setting Register (PPGn_PDUT) sets the duty of the PPG output waveform. In an 8-bit operation mode, PPGn_PDUT:PDUTH defines the duty of the PPGB output and PPGn_PDUT:PDUTL sets the duty of the PPGA signal.

PPG Duty Setting Register (PPGn_PDUT)

Figure 27-19. PPG Duty Setting Register (PPGn_PDUT)

PPGn_PDUT															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PDUTH[7]	PDUTH[6]	PDUTH[5]	PDUTH[4]	PDUTH[3]	PDUTH[2]	PDUTH[1]	PDUTH[0]	PDUTL[7]	PDUTL[6]	PDUTL[5]	PDUTL[4]	PDUTL[3]	PDUTL[2]	PDUTL[1]	PDUTL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 27-20. PPG Duty Setting Register (PPGn_PDUT)

Bit Position	Bit Field Name	Bit Description
[15:8]	PDUTH	PPG Duty Setting Register The PPG Duty Setting Register (PPGn_PDUT) sets the duty of the PPG output waveform. In 8-bit operation mode PDUTH defines the duty of the PPGB output. The PPG Duty Setting Register is buffered. Transfers from the buffers to the counter take place automatically upon counter borrow. Reading this register returns the buffered value i.e. actual duty setting. Set a smaller value in PPGn_PDUT than that which is set in the PPG Cycle Setting Register (i.e. PPGn_PCSR). If the same value is set in PPGn_PDUT as is set in PPGn_PCSR then: ■ Level 'H' is always the output at normal polarity (PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2 = '0'). ■ Level 'L' is always the output at inverted polarity (PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2 = '1').

Table 27-20. PPG Duty Setting Register (PPGn_PDUT)

Bit Position	Bit Field Name	Bit Description
[7:0]	PDUTL	PPG Duty Setting Register The PPG Duty Setting Register (i.e. PPGn_PDUT) sets the duty of the PPG output waveform. In 8-bit operation mode the PDUTL sets the duty of the PPGA signal. The PPG Duty Setting Register is buffered. Transfers from the buffers to the counter take place automatically upon counter borrow. Reading the register returns the buffered value i.e. actual duty setting. Set a smaller value in PPGn_PDUT than that which is set in the PPG Cycle Setting Register (i.e. PPGn_PCSR). If the same value is set in PPGn_PDUT as is set in PPGn_PCSR, then: ■ Level 'H' is always the output at normal polarity (PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2 = '0'). ■ Level 'L' is always the output at inverted polarity (PPGn_OPTMSK:OSEL/PPGn_OPTMSK:OSEL2 = '1').

27.2.18 PPG Timer Register (PPGn_PTMR)

The PPG Timer Register (PPGn_PTMR) reads the counts of PPGn. In an 8-bit operation mode PPGn_PTMR:PTMRH defines the PPG counter dedicated to the PPGB output and PPGn_PTMR:PTMRL reflects the PPG counter dedicated to the PPGA signal.

PPG Timer Register (PPGn_PTMR)

Figure 27-20. PPG Timer Register (PPGn_PTMR)

PPGn_PTMR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PTMRH[7]	PTMRH[6]	PTMRH[5]	PTMRH[4]	PTMRH[3]	PTMRH[2]	PTMRH[1]	PTMRH[0]	PTMRL[7]	PTMRL[6]	PTMRL[5]	PTMRL[4]	PTMRL[3]	PTMRL[2]	PTMRL[1]	PTMRL[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 27-21. PPG Timer Register (PPGn_PTMR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	PTMRH	PPG Timer Register Upper 8-bits The PPG Timer Register (PPGn_PTMR) reads the PPGn counts. In 8-bit operation mode, PTMRH defines the PPG counter dedicated to the PPGB signal.
[7:0]	PTMRL	PPG Timer Register 16-bit or Lower 8-bits The PPG Timer Register (PPGn_PTMR) reads the PPGn counts. In 8-bit operation mode, PTMRL defines the PPG counter dedicated to the PPGA signal.

27.2.19 PPG Start Delay Register (PPGn_PSDR)

The PPG Start Delay Register (PPGn_PSDR) controls delay of PWM output generation by PPGn_PSDR+1 cycles of PPG counter clock, when the PPGn_STRD:STRD bit is set to '1'. In the full range mode PWM output delay is PPGn_PSDR PPG count cycles. In an 8-bit operation mode PPGn_PSDR:PSDRH defines the start delay of the PPGB output and PPGn_PSDR:PSDRL sets the start delay of the PPGA signal.

PPG Start Delay Register (PPGn_PSDR)

Figure 27-21. PPG Start Delay Register (PPGn_PSDR)

PPGn_PSDR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PSDRH[7]	PSDRH[6]	PSDRH[5]	PSDRH[4]	PSDRH[3]	PSDRH[2]	PSDRH[1]	PSDRH[0]	PSDRL[7]	PSDRL[6]	PSDRL[5]	PSDRL[4]	PSDRL[3]	PSDRL[2]	PSDRL[1]	PSDRL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-22. PPG Start Delay Register (PPGn_PSDR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	PSDRH	PPG Start Delay Register The PPG Start Delay Register (PPGn_PSDR) controls the delay of the PWM output generation by PPGn_PSDR + 1 cycles of the PPG counter clock when the PPGn_STRD:STRD bit is set to '1'. In 8-bit operation mode the PSDRH defines the start delay of the PPGB output. In the full range mode the PWM output delay is PPGn_PSDR PPG count cycles. The PPGn_PSDR value should be set when PPG module operation is disabled (i.e. PPGn_CNTEN:CNTE = '0').
[7:0]	PSDRL	PPG Start Delay Register The PPG Start Delay Register (PPGn_PSDR) controls the delay of the PWM output generation by PPGn_PSDR + 1 cycles of the PPG counter clock when the PPGn_STRD:STRD bit is set to '1'. In 8-bit operation mode PSDRL sets the start delay of the PPGA signal. In the full range mode the PWM output delay is PPGn_PSDR PPG count cycles. The PPGn_PSDR value should be set when the PPG module operation is disabled (i.e. PPGn_CNTEN:CNTE = '0').

27.2.20 PPG Timing Point Capture Register (PPGn_PTPC)

The PPG Timing Point Capture Register (PPGn_PTPC) sets the timing point within the PWM cycle, which can be captured to generate an interrupt request or an ADC trigger if adequate control bits PPGn_EPCN1:TPCH/TPCL are set to '1'. In 8-bit operation mode PPGn_PTPC:PTPCH defines the timing point for the PPGB output and PPGn_PTPC:PTPCL sets the timing point for the PPGA signal.

PPG Timing Point Capture Register (PPGn_PTPC)

Figure 27-22. PPG Timing Point Capture Register (PPGn_PTPC)

PPGn_PTPC															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PTPCH[7]	PTPCH[6]	PTPCH[5]	PTPCH[4]	PTPCH[3]	PTPCH[2]	PTPCH[1]	PTPCH[0]	PTPCL[7]	PTPCL[6]	PTPCL[5]	PTPCL[4]	PTPCL[3]	PTPCL[2]	PTPCL[1]	PTPCL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-23. PPG Timing Point Capture Register (PPGn_PTPC) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	PTPCH	PPG Timing Point Capture Register The PPG Timing Point Capture Register (PPGn_PTPC) sets the timing point within the PWM cycle that can be captured to generate an interrupt request or an ADC trigger if adequate control bits PPGn_EPCN1:TPCH/PPGn_EPCN1:TPCL are set to '1'. In 8-bit operation mode PTPCH defines the timing point for the PPGB output.
[7:0]	PTPCL	PPG Timing Point Capture Register The PPG Timing Point Capture Register (PPGn_PTPC) sets the timing point within the PWM cycle that can be captured to generate an interrupt request or an ADC trigger if adequate control bits PPGn_EPCN1:TPCH/PPGn_EPCN1:TPCL are set to '1'. In 8-bit operation mode PTPCL sets the timing point for the PPGA signal.

27.2.21 PPG End Duty Register (PPGn_PEDR)

The PPG End Duty Register (PPGn_PEDR) sets the end point of the PWM duty ramp in ramp operation mode PPGn_RMPCFG:RAMPH/RAMPL = '1'. In 8-bit operation mode PPGn_PEDR:PEDRH defines the end duty of the PPGB output and PPGn_PEDR:PEDRL sets the end duty of the PPGA signal.

PPG End Duty Register (PPGn_PEDR)

Figure 27-23. PPG End Duty Register (PPGn_PEDR)

PPGn_PEDR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PEDRH[7]	PEDRH[6]	PEDRH[5]	PEDRH[4]	PEDRH[3]	PEDRH[2]	PEDRH[1]	PEDRH[0]	PEDRL[7]	PEDRL[6]	PEDRL[5]	PEDRL[4]	PEDRL[3]	PEDRL[2]	PEDRL[1]	PEDRL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-24. PPG End Duty Register (PPGn_PEDR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	PEDRH	PPG End Duty Register The PPG End Duty Register sets the end point of the PWM duty ramp in ramp operation mode PPGn_RMPCFG:RAMPH/ PPGn_RMPCFG:RAMPL = '1'. In 8-bit operation mode PEDRH defines the 'end duty' of the PPGB output. In the ramp mode PWM duty is incremented (i.e. PPGn_RMPCFG:RIDH/PPGn_RMPCFG:RIDL = '0') or decremented (i.e. PPGn_RMPCFG:RIDH/PPGn_RMPCFG:RIDL = '1') with every pulse of the selected input signal. the PWM output waveform starts with the duty defined by PPGn_PDUT register and the duty value is incremented/decremented until the last (i.e. end) value is reached.

Table 27-24. PPG End Duty Register (PPGn_PEDR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	PEDRL	PPG End Duty Register The PPG End Duty Register sets the end point of the PWM duty ramp in ramp operation mode, i.e. PPGn_RMPCFG:RAMPL = '1'. In 8-bit operation mode PEDRL defines the 'end duty' of the PPGA signal. In ramp mode the PWM duty is incremented (i.e. PPGn_RMPCFG:RIDL = '0') or decremented (i.e. PPGn_RMPCFG:RIDL = '1') with every pulse of the selected input signal. The PWM output waveform starts with the duty defined by the PPGn_PDUT register and the duty value is incremented/decremented until the last (i.e. end) value is reached.

27.2.22 PPG DMA Configuration Register (PPGn_DMACFG)

The PPG DMA Configuration Register (PPGn_DMACFG) enables the DMA request for the PPG module.

PPG DMA Configuration Register (PPGn_DMACFG)

Figure 27-24. PPG DMA Configuration Register (PPGn_DMACFG)

PPGn_DMACFG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	ENDMAREQ
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 27-25. PPG DMA Configuration Register (PPGn_DMACFG) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	ENDMAREQ	Enable DMA Request '0': No DMA request '1': Enable DMA request

27.2.23 PPG Debug Enable Register (PPGn_DEBUG)

The PPG Debug Enable Register (PPGn_DEBUG) enables the debug mode for the PPG module.

PPG Debug Enable Register (PPGn_DEBUG)

Figure 27-25. PPG Debug Enable Register (PPGn_DEBUG)

PPGn_DEBUG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	DBGEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 27-26. PPG Debug Enable Register (PPGn_DEBUG) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	DBGEN	PPG Debug Enable This bit is used to enable/disable debug mode for the PPG. '0': Disable debug mode '1': Enable debug mode When DBGEN is set to '1' and the processor is in debug state, the PPG prescaler is stopped. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.

27.2.24 Group Control Register (PPGGRPp_GCTRL)

The Group Control Register (PPGGRPp_GCTRL) generates internal trigger levels using software for a PPG block of four PPGs.

Group Control Register (PPGGRPp_GCTRL)

Figure 27-26. Group Control Register (PPGGRPp_GCTRL)

PPGGRPp_GCTRL							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	EN[3]	EN[2]	EN[1]	EN[0]
Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-27. Group Control Register (PPGGRPp_GCTRL) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3:0]	EN	Internal Triggers Sets the levels of internal triggers. '0': Sets the level to 'L' '1': Sets the level to 'H' If any of the EN trigger inputs (EN[3:0]) is selected with the trigger specification bits (i.e. PPGn_GCN1:TSEL[3:0]) of PPG, then the selected EN serves as a PPG trigger input bit. If the state selected with the trigger input edge selection bit (i.e. PPGn_PCN:EGS[1:0]) is generated by software using the trigger input bit (selected EN[0], EN[1], EN[2] or EN[3]), the choice serves as an activation trigger to activate the PPG.

27.2.25 PPG Global Control Register (PPGGLCg_GCNR)

The PPG Global Control Register (PPGGLCg_GCNR) is common for all PPG groups on the same peripheral bus. Two bits CTG[0] and CTG[1] can be used as common triggers for many or all on-chip available PPG groups on the same peripheral bus.

PPG Global Control Register (PPGGLCg_GCNR)

Figure 27-27. PPG Global Control Register (PPGGLCg_GCNR)

PPGGLCg_GCNR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	CTG[1]	CTG[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 27-28. PPG Global Control Register (PPGGLCg_GCNR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1:0]	CTG	Common Trigger These bits set the level of the common triggers CTG[0] and CTG[1]. '0': Sets the level to 'L' '1': Sets the level to 'H' If either CTG[0] or CTG[1] is selected with the trigger specification bits (i.e. PPGn_GCN1:TSEL[3:0]), then the selected CTG serves as a PPG trigger input bit. If the state selected with the trigger input edge selection bits (i.e. PPGn_PCN:EGS[1:0]) is generated by software using either CTG[0] or CTG[1], the choice serves as a trigger to activate the PPG.

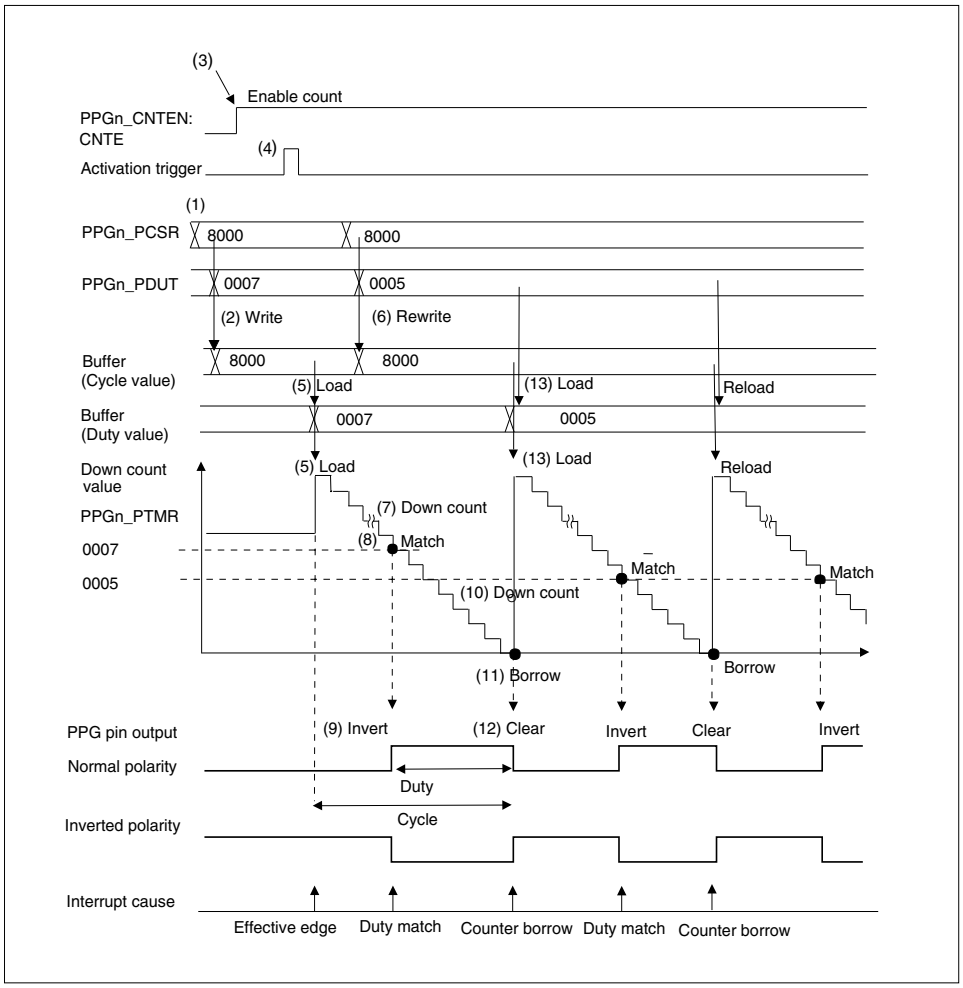
27.3 Operation of the Programmable Pulse Generator

The PPGs provide a programmable pulse output independently or jointly. This section describes the individual modes of operation.

PWM operation

In PWM operation, variable duty pulses are generated at the PPG pin.

Figure 27-28. PWM operation



1. Write a cycle value.
2. Write a duty value and transfer the cycle value to the buffers.
3. Enable PPG operation.
4. Generate an activation trigger.
5. Load the cycle value to the counter and the duty value to the buffer.
6. Rewrite the duty value and transfer the cycle value to the buffers.
7. The counter counts down.
8. The down counter equals the duty value.
9. Invert the PPG pin output level.
10. Counter counts down.

11. Counter borrow.
12. Clear the PPG pin output level (return to normal).
13. Reload the cycle value to the counter and the duty value to the buffer.
14. Steps (7) to (13) are reiterated.

Equations:

Period = (PPGn_PCSR +1) x count clock Duty = (PPGn_PDUT +1) x count clock

Width up to pulse out = (PPGn_PCSR - PPGn_PDUT) x Countclock

where

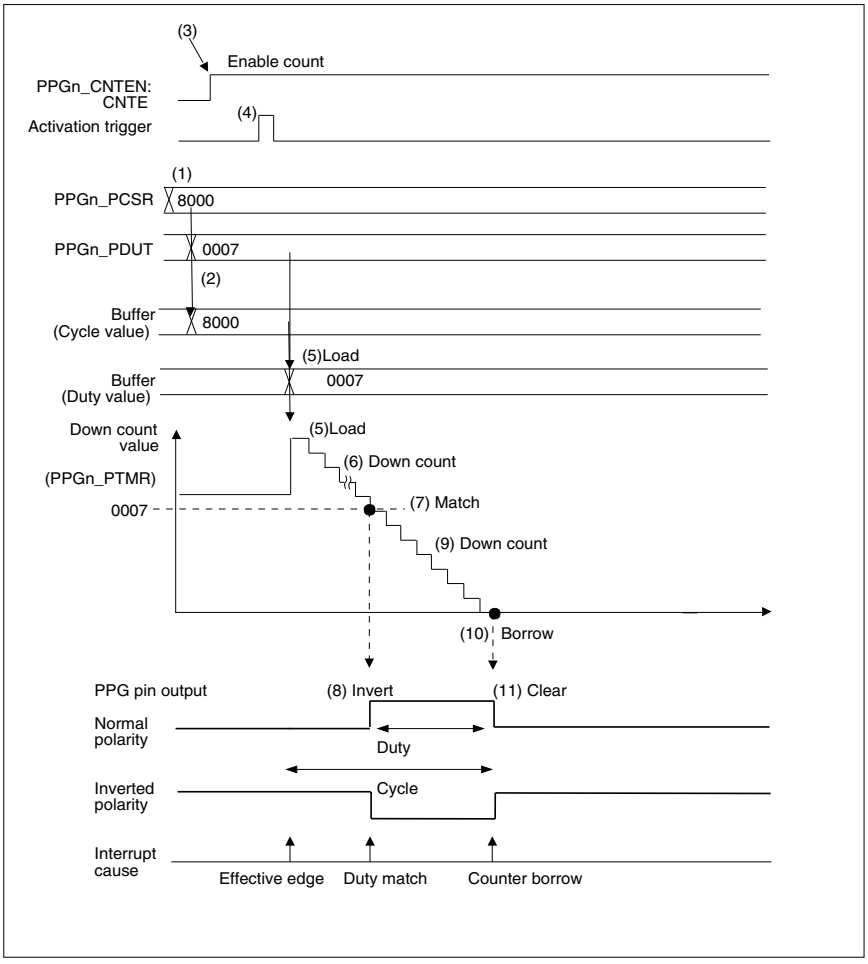
PPGn_PCSR is the cycle value

PPGn_PDUT is the duty value

One-shot operation

In one-shot operation, one-shot pulses are generated at the PPG pin. One-shot operation cannot be used in 8-bit mode or together with the start delay feature.

Figure 27-29. One-shot operation

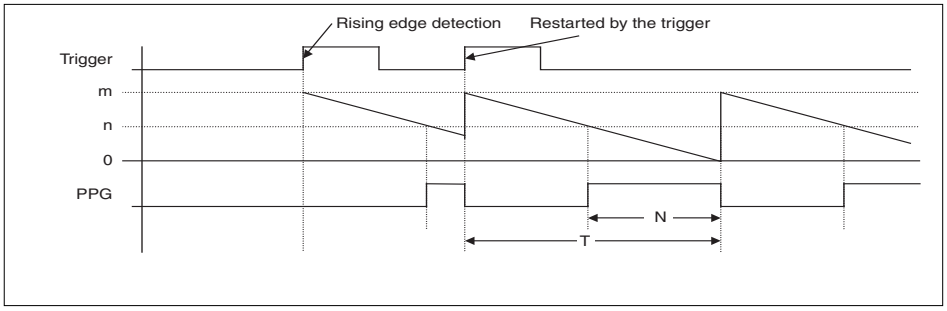


1. Write a cycle value.
2. Write a duty value and transfer the cycle value to the buffers.

3. Enable PPG operation.
4. Generate an activation trigger.
5. Load the cycle value to the counter and the duty value to the buffer.
6. The counter counts down.
7. The down counter equals the duty value.
8. Invert the PPG pin output level.
9. The counter counts down.
10. Counter borrow.
11. Clear the PPG pin output level (return to normal).
12. The operation sequence is now complete.

Restart operation

Figure 27-30. Restart available in PWM operation



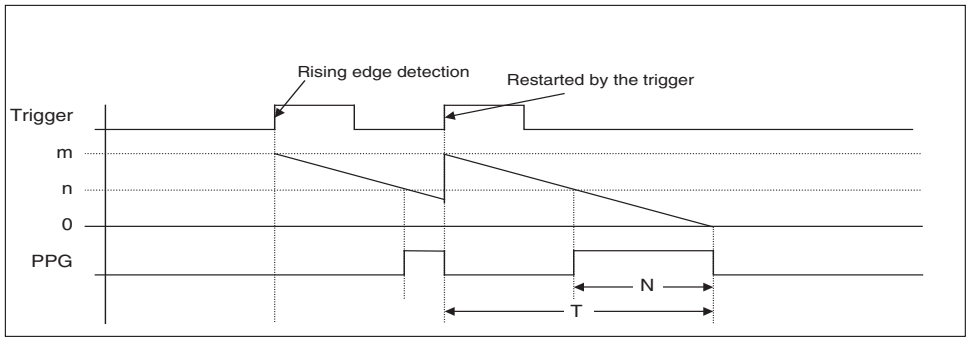
m = cycle value

n = duty value

T = cycle time

N = duty time

Figure 27-31. Restart available in one-shot operation



m = cycle value

n = duty value

T = cycle time

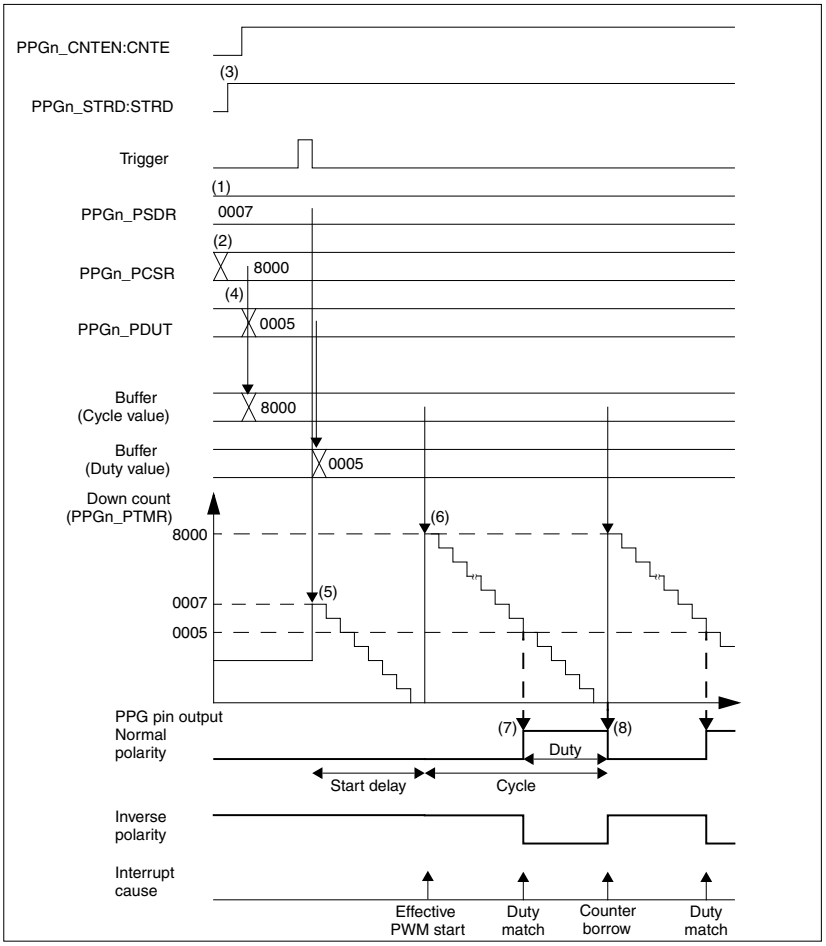
N = duty time

If a restart is not available, the second and subsequent triggers have no effect in both PWM and one-shot operation (the second and subsequent triggers following a shut down of the down counter are functional).

Start delay mode

In start delay mode, the generation of the PWM output is postponed by PPGn_PSDR PPG count cycles.

Figure 27-32. Start delay mode

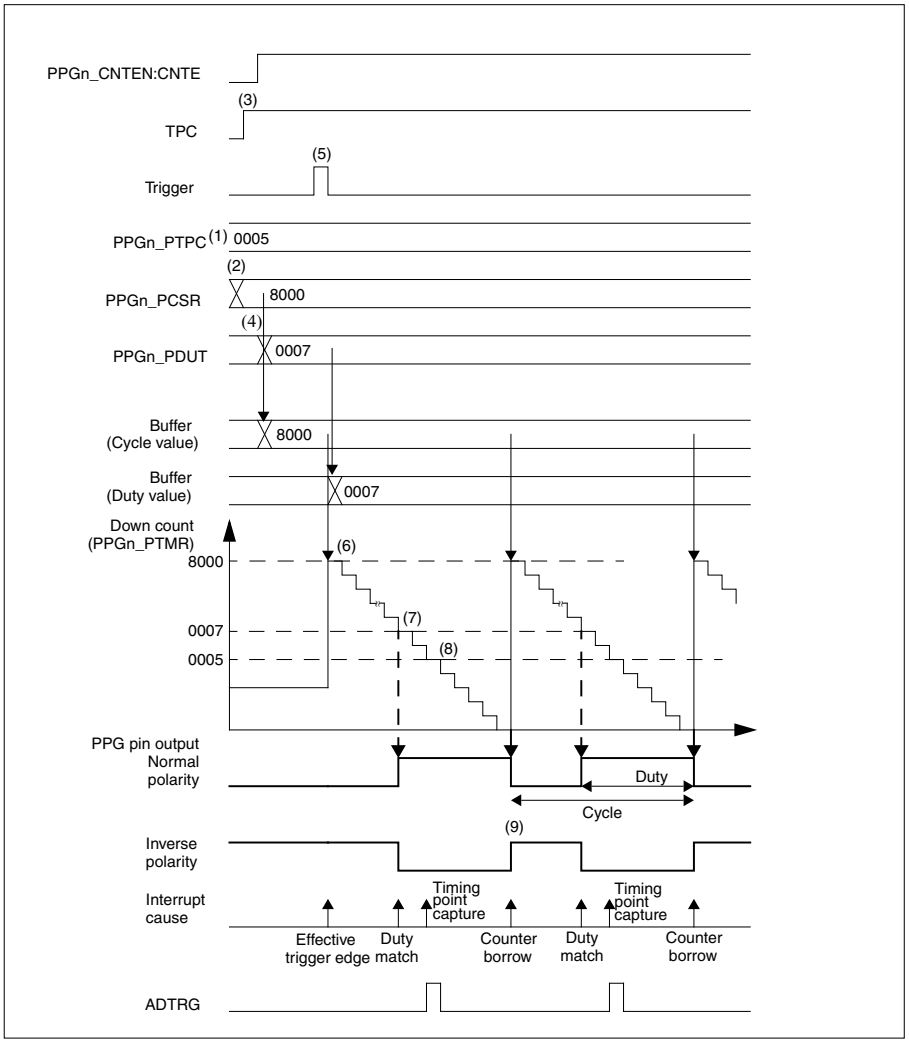


1. Write the start delay value.
2. Write the cycle value.
3. Enable start delay mode.
4. Write the duty value.
5. The PPG is triggered and starts operation. PPGn_PTMR is loaded with the value in PPGn_PSDR.
6. PPGn_PTMR is loaded with the PPGn_PCSR (cycle) value after the start delay phase.
7. The PPG output toggles after PPGn_PTMR matches the duty value.
8. The PPG output toggles after PPGn_PTMR reaches zero, and the PPGn_PTMR is loaded with the cycle value.

Timing point capture mode

In this operation mode a timing point within the PWM cycle can be defined, which is used as a possible interrupt request or ADC trigger signal.

Figure 27-33. Timing point capture mode

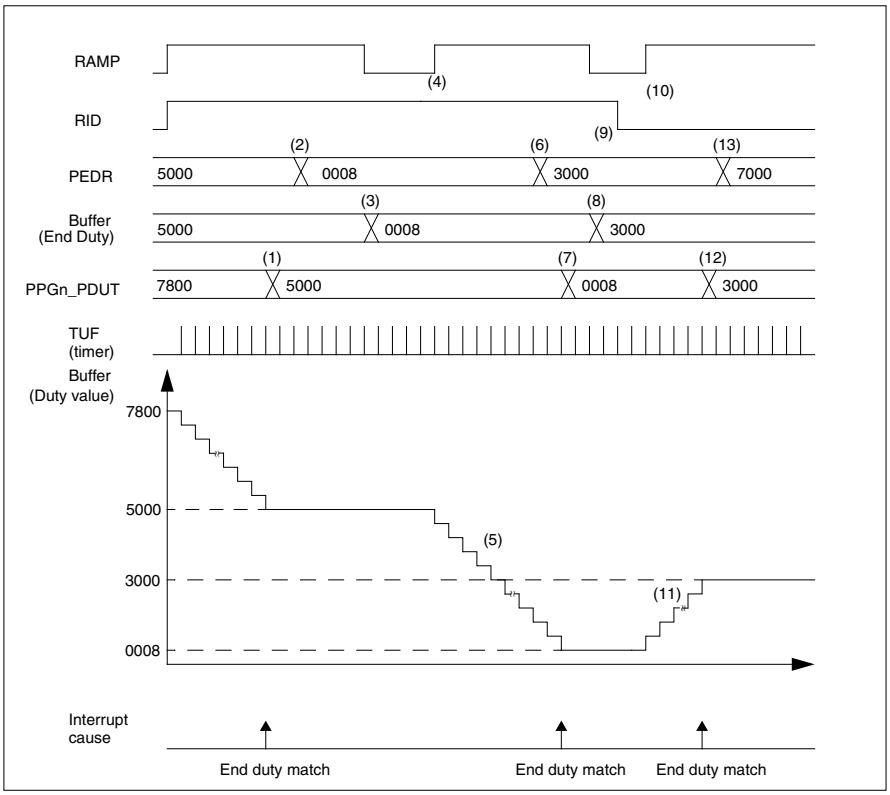


1. Write the timing point capture value.
2. Write the cycle value.
3. Enable the timing point capture mode.
4. Write the duty value.
5. Trigger the PPG.
6. PPGn_PTMR is loaded with cycle value and starts counting down.
7. Duty match occurs when PPGn_PTMR matches with duty value and the output of the PPG toggles.
8. PPGn_PTMR matches the PPGn_PTPC register and an interrupt is generated if the PPGn_PCN:IRS bits are configured for timing point capture. Also a trigger signal is generated for ADC.
9. Counter borrow happens and the PPG output toggles.

Ramp mode

In ramp mode, the PWM duty is incremented (i.e. PPGn_RMPCFG:RIDH/PPGn_RMPCFG:RIDL = '0') or decremented (i.e. PPGn_RMPCFG:RIDH/PPGn_RMPCFG:RIDL = '1') with every pulse of the input signal selected by PPGn_GCN3. The PWM output waveform starts with the duty defined by the PPGn_PDUT register and the duty value is incremented/decremented until the end duty is reached.

Figure 27-34. Ramp mode

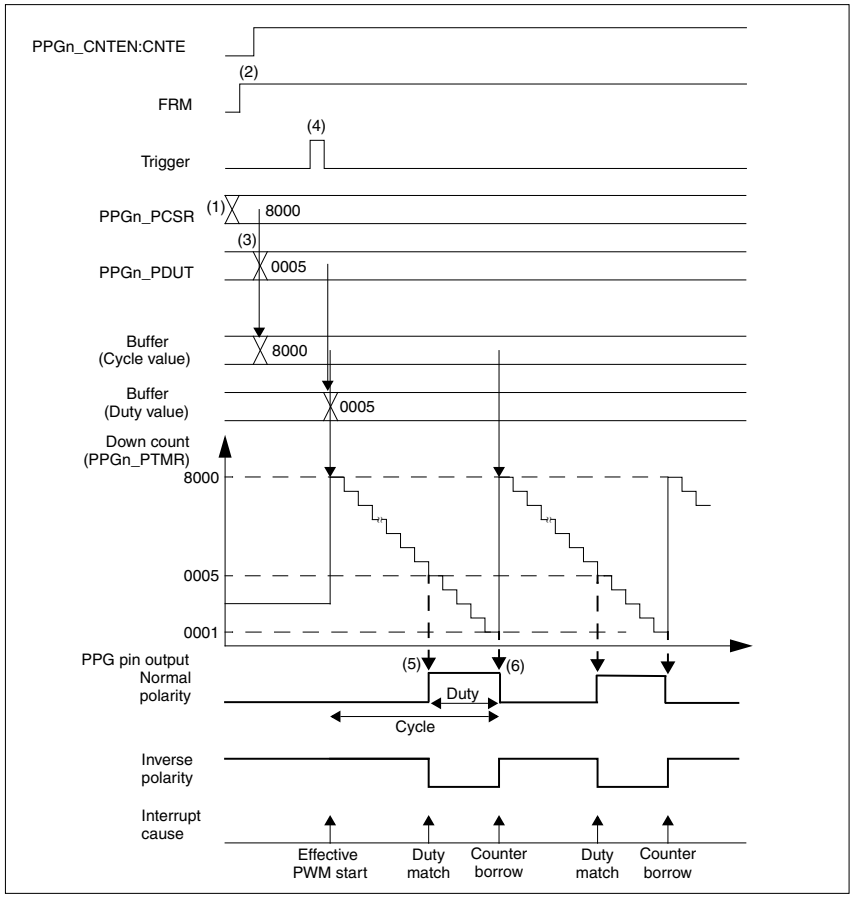


1. After the end duty is reached, the PPGn_PDUT register is updated to the end duty value.
2. The new PPGn_PEDR value is set by software.
3. Since ramp mode is disabled, the PPGn_PEDR value is transferred to the actual end duty.
4. Enable RAMP mode
5. Decrease the duty buffer value with each Timer Underflow (TUF)
6. The new PPGn_PEDR value is set by software.
7. The end duty is reached, and the PPGn_PDUT register is updated to the end duty value.
8. Ramp mode is disabled, hence the PPGn_PEDR value is transferred to the actual end duty.
9. Change RAMP direction to increase
10. Enable RAMP mode
11. Increase the duty buffer value with each Timer Underflow (TUF)
12. End duty is reached, and the PPGn_PDUT register is updated to the end duty value.
13. The new PPGn_PEDR value is set by software, but it becomes active (transferred to end duty buffer register) after ramp mode is disabled.

Full range mode

PPG counter borrow happens at PPGn_PTMR = '1'. As a consequence, to set PPGn_PDUT = '0', the PPG output pin stays '0' all the time and accordingly, to set PPGn_PDUT = PPGn_PCSR PPG, the output stays at a high level during the entire operation time.

Figure 27-35. Full range mode



Equations:

Period = (PPGn_PCSR +1) x count clock Duty = (PPGn_PDUT +1) x count clock where

PPGn_PCSR is the cycle value

PPGn_PDUT is the duty value

Width up to pulse out = (PPGn_PCSR - PPGn_PDUT) x count clock

1. Write the cycle value.
2. Enable full range mode.
3. Write the duty value.
4. The PPG is triggered, and the PPGn_PTMR is loaded with the cycle value and starts counting down.
5. PPGn_PTMR matches the duty value, and the PPG output toggles.
6. The PPG output toggles on counter borrow.

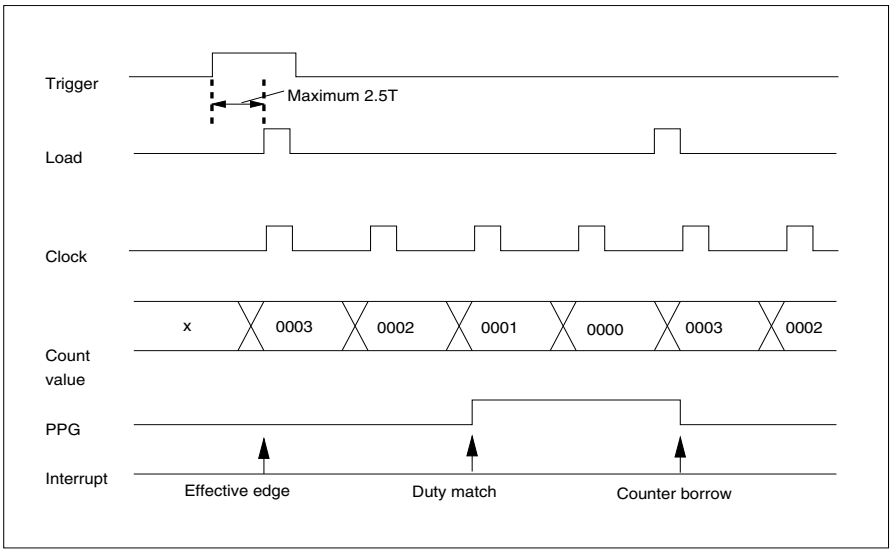
27.4 Cautions

This section describes the cautions to be considered while using the Programmable Pulse Generator.

Cautions

- If the interrupt request flag (PPGn_PCN:IRQF) equals '1' and the interrupt request flag is set to '0' at the same time, the setting of the interrupt request flag to '1' overrides the flag clear request
- The first load has a maximum delay of 2.5 T after the activation trigger (T: count clock period). If the down counter is loaded and counts at the same time, the load operation overrides the count operation with a new value from the buffer

Figure 27-36. Cautions while using PPG



- Ensure that the duty value is written to PPGn_PDUT after the cycle value has been initialized in PPGn_PCSR and rewritten. Only the PPGn_PDUT can be written to to rewrite the duty
- The duty value in PPGn_PDUT must be equal to or smaller than the cycle value in PPGn_PCSR. If a larger value has been set, disable the PPG operation before replacing the duty value with a smaller value
- To activate a PPG, it is necessary to set the timer operation enable bits (i.e. PPGn_CNTEN:CNTE) to '1' before (or concurrently as) PPG operation is activated
- The mode of operation selected (PPGn_PCN:MDSE), restart enable (PPGn_PCN:RTRG), count clock (PPGn_PCN:CKS[1:0]), trigger input edge (PPGn_PCN:EGS[1:0]), interrupt cause (PPGn_PCN:IRS[2:0]), internal trigger (PPGn_GCN1:TSEL[3:0]) and output polarity specification (PPGn_OPTMSK:OSEL) may not be changed during PPG operation. If any of these values has been changed during operating, disable the operation of the PPG before reloading the register
- If the activation trigger specification bits (i.e. PPGn_GCN1:TSEL[3:0]) have been set to any value outside the specified range ('1100' - '1111'), disable PPG operation and write the specified value to allow the register to return to normal
- If the timer operation enable bit (i.e. PPGn_CNTEN:CNTE) is set to '0' to disable the PPGn while it is operating, the PPG stops, its counter is latched and the PPG output level is reset to a default value (i.e. ‘L’ if PPGn_OPTMSK:OSEL = '0' or ‘H’ if PPGn_OPTMSK:OSEL = '1'). To reactivate PPG operation, PPGn_CNTEN:CNTE is set to '1' before or concurrently as a PPG trigger is provided to enable the PPG. Following this, the PPG counter is loaded with the PPG cycle value and PPG pulses are generated
- A one-shot pulse can be generated in 16-bit operation mode only and without a start delay feature
- The PPG output signals PPGAn and PPGBn are processed by a gating function before being output to the device pins. The gated PPGAn output is propagated to the PPGn_PPGA pin and the gated PPGBn output is routed to the PPGn_PPGB pin. Refer to 14. Resource Input Configuration for more information about how to configure these gates

28. 32-Bit Reload Timer

This chapter explains the function and operation of the 32-bit Reload Timer.

28.1 Outline of the 32-bit Reload Timer

This section describes the features and the block diagram of the 32-bit Reload Timer.

Features of the 32-bit Reload Timer

The 32-bit Reload Timer consists of a 32-bit down counter, a 32-bit Reload register, one input (TIN), one output (TOT) and control registers. The 32-bit Reload Timer has the following features:

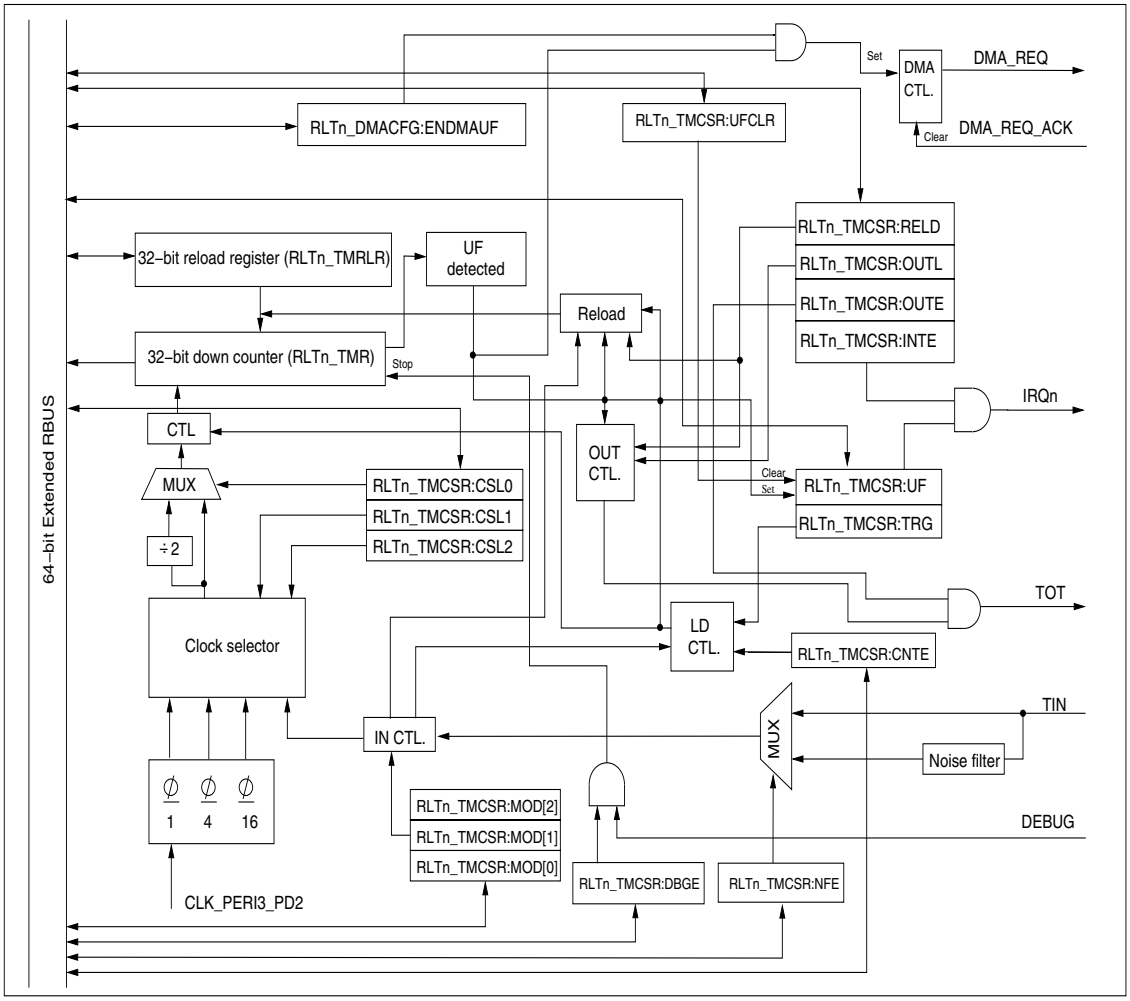
- An external and internal clock/event source
- A trigger signal that can be programmed as rising/falling edge or both
- A gated count function
- One-shot or reload counter mode
- A counter state that can be made visible at the external pin
- A prescaler with six different settings for the internal clock and two different settings for the external clock
- Several Reload Timers can be cascaded to form a longer Reload Timer
- Support for the Microcontroller (MCU) debug mode

DMA and interrupts

The 32-bit Reload Timer can generate a DMA request which can be used to start DMA transfers. The 32-bit Reload Timer can generate an interrupt in the event of an underflow.

Block diagram of 32-bit Reload Timer

Figure 28-1. Block diagram of 32-bit Reload Timer



Note: For more details about cascading the Reload Timer, refer to 14. Resource Input Configuration

28.2 32-bit Reload Timer registers

The registers of the 32-bit Reload Timer are explained in this section.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of 32-bit Reload Timer

The following registers are available for each instance of the 32-bit Reload Timer:

- DMA Configuration Register (RLTn_DMACFG)
- Timer Control Status Register (RLTn_TMCSR)
- 32-bit Reload Register (RLTn_TMRLR)
- 32-bit Timer Register (RLTn_TMR)

Memory layout of 32-bit Reload Timer registers

Table 28-1. Memory layout of 32-bit Reload Timer registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				RLTn_DMACFG XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000008	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				RLTn_TMCSR 00000000 00000000 00000000 00000000			
0x00000010	RLTn_TMR XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				RLTn_TMRLR XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			

Note: For more details on protection feature, refer to 21. Peripheral Protection Unit.

28.2.1 DMA Configuration Register (RLTn_DMACFG)

The DMA Configuration Register controls the DMA request generation for an underflow condition.

DMA Configuration Register (RLTn_DMACFG)

Figure 28-2. DMA Configuration Register (RLTn_DMACFG)

RLTn_DMACFG																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENDMAUF
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 28-2. DMA Configuration Register (RLTn_DMACFG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENDMAUF	DMA Enable for Underflow '0': No DMA request is generated '1': A DMA request is generated when the counter, RLTn_TMR, underflows

28.2.2 Timer Control Status Register (RLTn_TMCSR)

The Timer Control Status Register controls the operation mode and interrupt of the 32-bit Reload Timer.

Timer Control Status Register (RLTn_TMCSR)

Figure 28-3. Timer Control Status Register (RLTn_TMCSR)

RLTn_TMCSR																															
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**
read0	read0	read0	read0	read0	read0	read0	CNTE	read0	read0	read0	read0	read0	TRG	UFCLR	UF	MOD[2]	MOD[1]	MOD[0]	CSL2	CSL1	CSL0	read0	NFE	DBGE	OUTE	OUTL	RELD	INTE	read0	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-3. Timer Control Status Register (RLTn_TMCSR) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	CNTE	Count Enable This bit is a timer count enable bit. '0': Stops count operation '1': Sets the timer to wait for a trigger
[23:19]	read0	-
[18]	TRG	Trigger Software trigger bit. '0': No effect '1': A software trigger is applied, causing the timer to load the reload register content to the counter and start counting Reading this bit always returns '0'. **Notes:** 1. Applying a trigger using this register is only valid when RLTn_TMCSR:CNTE = '1'. If RLTn_TMCSR:CNTE = '0', writing '1' to TRG has no effect. 2. Set this bit in 'Gate Input mode' to load the reload register content before counting starts.
[17]	UFCLR	Underflow Interrupt Clear 0': No effect '1': Clears the RLTn_TMCSR:UF bit Reading this bit always returns '0'.

Table 28-3. Timer Control Status Register (RLTn_TMCSR) bits

Bit Position	Bit Field Name	Bit Description
[16]	UF	Underflow This read-only bit is the timer interrupt request flag. '0': No underflow has occurred '1': An underflow has occurred The UF bit is cleared by writing '1' to the RLTn_TMCSR:UFCLR bit.
[15:13]	MOD	Operation Mode These bits set the operation mode and input (TIN) functions. The MOD[2] bit selects the input (TIN) function. '0': The input TIN is used as a trigger input. In this case, the Reload Register content is loaded to the counter when an active edge is input to the TIN and count operation proceeds '1': The timer operates in gated counter mode and the input TIN is used as a gate input. In this mode, the counter only counts while an active level is input to the input TIN Table 28-4 and Table 28-5 list the MOD[2:0] bit settings.
[12]	CSL2	Clock Select 2 This bit specifies the clock/event source and the clock division ratio. Table 28-6 lists the selected clock sources for different CSL0/CSL1/CSL2 settings.
[11]	CSL1	Clock Select 1 This bit specifies the clock/event source and the clock division ratio. Table 28-6 lists the selected clock sources for different CSL0/CSL1/CSL2 settings.
[10]	CSL0	Clock Select 0 This bit specifies the count clock division ratio. '0': The count clock specified by the count clock selection bits, RLTn_TMCSR:CSL2 and RLTn_TMCSR:CSL1, is not divided '1': The count clock specified by the count clock selection bits, RLTn_TMCSR:CSL2 and RLTn_TMCSR:CSL1, is divided by two Table 28-6 lists the selected clock sources for different CSL0/CSL1/CSL2 settings.
[9]	read0	-
[8]	NFE	Noise Filter Enable This bit is used to enable/disable the noise filter for the TIN input. '0': Noise filter for TIN is disabled '1': Noise filter for TIN is enabled

Table 28-3. Timer Control Status Register (RLTn_TMCSR) bits

Bit Position	Bit Field Name	Bit Description
[7]	DBGE	Debug Mode Enable This bit is used to enable/disable debug mode for the RLT. '0': Debug mode disabled '1': Debug mode enabled When DBGE is set to '1' and the processor is in debug state, the timer counter operation is paused and writing to the RLTn_TMRLR register directly updates the timer counter (RLTn_TMR register). When the processor leaves debug state or DBGE is set to '0', the timer counter operation is resumed. For the definition of debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[6]	OUTE	Output Enable '0': The timer output TOT is disabled '1': TOT is used as the Reload Timer output Refer to Table 28-7 which outlines the settings for the RLTn_TMCSR:OUTE, RLTn_TMCSR:OUTL and RLTn_TMCSR:RELD bits.
[5]	OUTL	Output Level This bit sets the output level for the TOT. Refer to Table 28-7 which outlines the settings for the RLTn_TMCSR:OUTE, RLTn_TMCSR:OUTL and RLTn_TMCSR:RELD bits.
[4]	RELD	Reload This bit enables reload operations. '0': The timer operates in one-shot mode. In this mode, the count operation stops when an underflow occurs due to the counter value changing from 0x00000000 to 0xFFFFFFFF '1': The timer operates in reload mode. In this mode, the timer loads the reload register contents into the counter and continues counting whenever an underflow occurs Refer to Table 28-7 which outlines the settings for the RLTn_TMCSR:OUTE, RLTn_TMCSR:OUTL and RLTn_TMCSR:RELD bits.
[3]	INTE	Interrupt Enable Timer interrupt request enable bit. '0': No interrupt request is generated even when the RLTn_TMCSR:UF bit changes to '1' '1': An interrupt request is generated when the RLTn_TMCSR:UF bit changes to '1'
[2:0]	read0	-

Table 28-4. RLTn_TMCSR:MOD[2:0] bit settings for the internal clock mode (RLTn_TMCSR:CSL1 /RLTn_TMCSR:CSL2 = '00', '01', or '10')

RLTn_TMCSR :MOD[2]	RLTn_TMCSR :MOD[1]	RLTn_TMCSR :MOD[0]	Input function	Active edge or level
'0'	'0'	'0'	Trigger disabled	-
'0'	'0'	‘1’	Trigger input	Rising edge
'0'	‘1’	'0'		Falling edge
'0'	‘1’	'1'		Both edges
'1'	x	'0'	Gate input	'L' level
'1'	x	'1'		'H' level

Table 28-5. RLTn_TMCSR:MOD[2:0] bit settings for event counter mode (RLTn_TMCSR:CSL1 /RLTn_TMCSR:CSL2 = '11')

RLTn_TMCSR :MOD[2]	RLTn_TMCSR: MOD[1]	RLTn_TMCSR: MOD[0]	Input function	Active edge or level
x	'0'	'0'	-	-
	'0'	'1'	Event input	Rising edge
	'1'	'0'		Falling edge
	'1'	'1'		Both edges

Note: Bits marked as 'x' in the table can be set to any value.

Table 28-6. Clock sources for RLTn_TMCSR:CSL0 /RLTn_TMCSR:CSL1 /RLTn_TMCSR:CSL2 bit settings

RLTn_TMCSR: CSL2	RLTn_TMCSR: CSL1	RLTn_TMCSR: CSL0	Count clock (time for peripheral clock)
'0'	'0'	'0'	CLK_PERI3_PD2/1
'0'	'0'	'1'	CLK_PERI3_PD2/2
'0'	'1'	'0'	CLK_PERI3_PD2/4
'0'	'1'	'1'	CLK_PERI3_PD2/8
'1'	'0'	'0'	CLK_PERI3_PD2/16
'1'	'0'	'1'	CLK_PERI3_PD2/32
'1'	'1'	'0'	External event count mode, each event on TIN
'1'	'1'	'1'	External event count mode, each second event on TIN

Table 28-7. RLTn_TMCSR:OUTE, RLTn_TMCSR:OUTL, and RLTn_TMCSR:RELD settings

RLTn_TMCSR: OUTE	RLTn_TMCSR: OUTL	RLTn_TMCSR: RELD	Output waveform
'0'	x	x	Timer output disabled
'1'	'0'	'0'	Output an 'H' level pulse during counting.
'1'	'1'	'0'	Output an 'L' level pulse during counting.
'1'	'0'	'1'	Toggle output. Starts with 'L' level output. Changes level on timer reload.
'1'	'1'	'1'	Toggle output. Starts with 'H' level output. Changes level on timer reload.

28.2.3 32-bit Reload Register (RLTn_TMRLR)

The 32-bit Reload Register holds the reload value. The initial value is undefined.

32-bit Reload Register (RLTn_TMRLR)

Figure 28-4. 32-bit Reload Register (RLTn_TMRLR)

RLTn_TMRLR																															
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
TMRLR[31]	TMRLR[30]	TMRLR[29]	TMRLR[28]	TMRLR[27]	TMRLR[26]	TMRLR[25]	TMRLR[24]	TMRLR[23]	TMRLR[22]	TMRLR[21]	TMRLR[20]	TMRLR[19]	TMRLR[18]	TMRLR[17]	TMRLR[16]	TMRLR[15]	TMRLR[14]	TMRLR[13]	TMRLR[12]	TMRLR[11]	TMRLR[10]	TMRLR[9]	TMRLR[8]	TMRLR[7]	TMRLR[6]	TMRLR[5]	TMRLR[4]	TMRLR[3]	TMRLR[2]	TMRLR[1]	TMRLR[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 28-8. 32-bit Reload Register (RLTn_TMRLR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TMRLR	Timer Reload Register This 32-bit reload register holds the reload value. The initial value is undefined. RLTn_TMRLR can only be accessed in 32- or 64-bit mode. When RLTn_TMCSR:DBGE is set to '1' and the processor is in debug state, writing to this register updates the timer counter immediately.

28.2.4 32-bit Timer Register (RLTn_TMR)

Reading this register returns the count value of the 32-bit Reload Timer. The initial value is undefined.

32-bit Timer Register (RLTn_TMR)

Figure 28-5. 32-bit Timer Register (RLTn_TMR)

RLTn_TMR																															
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**
TMR[31]	TMR[30]	TMR[29]	TMR[28]	TMR[27]	TMR[26]	TMR[25]	TMR[24]	TMR[23]	TMR[22]	TMR[21]	TMR[20]	TMR[19]	TMR[18]	TMR[17]	TMR[16]	TMR[15]	TMR[14]	TMR[13]	TMR[12]	TMR[11]	TMR[10]	TMR[9]	TMR[8]	TMR[7]	TMR[6]	TMR[5]	TMR[4]	TMR[3]	TMR[2]	TMR[1]	TMR[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 28-9. 32-bit Timer Register (RLTn_TMR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TMR	Timer Register Reading this register returns the count value of the 32-bit Reload Timer. The initial value is undefined. RLTn_TMR can only be accessed in 32- or 64-bit mode. When RLTn_TMCSR:DBGE is set to '1' and the processor is in debug state, writing to this register updates the timer counter immediately.

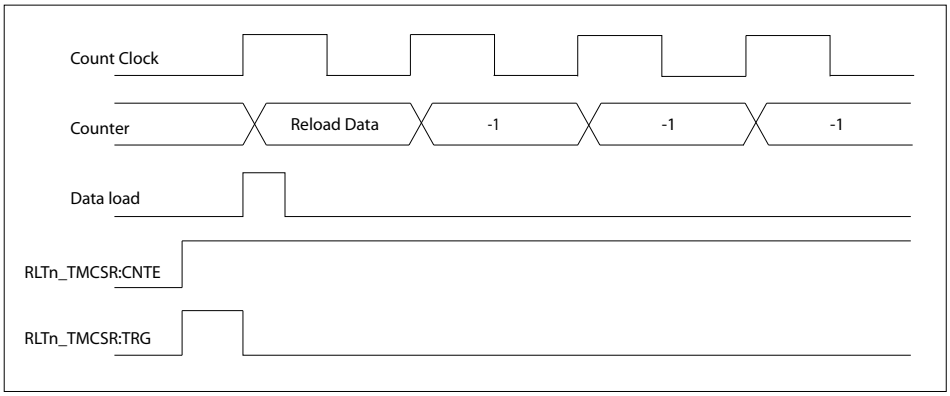
28.3 Internal clock and external event counter operation of the 32-bit Reload Timer

In internal clock mode, the peripheral clock with different divider settings can be selected as the clock source to operate the Reload Timer. The external input TIN can be selected as either a trigger input or as a gate input by a register setting. In event counter mode, the TIN is used as an external event input. Each active edge on this input (rising, falling or both) decrements the counter. When RLTn_TMCSR:CSL0 = '1', then every second event is counted.

Internal clock operation of the 32-bit Reload Timer

Writing '1' to the RLTn_TMCSR:CNTE and RLTn_TMCSR:TRG bits both enables and starts counting at the same time. Using the RLTn_TMCSR:TRG bit as a trigger input is always possible when the timer is enabled (RLTn_TMCSR:CNTE = '1'), regardless of the operation mode. Figure 28-6 shows counter activation and operation.

Figure 28-6. Activation and operation of the 32-bit Reload Timer counter



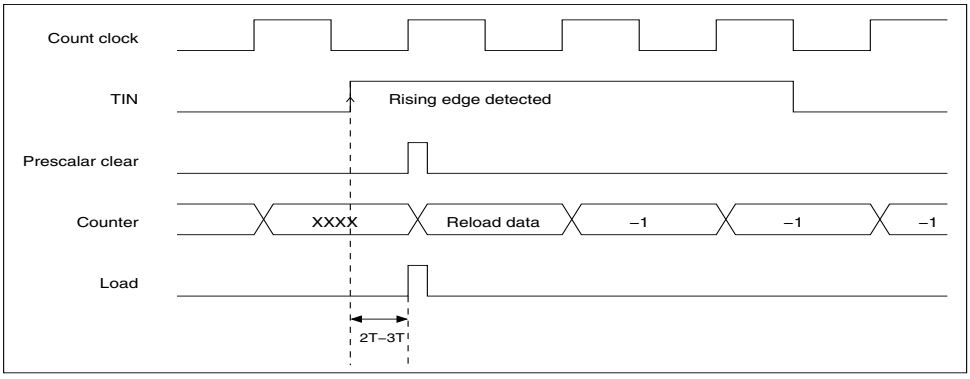
Input functions of the 32-bit Reload Timer (in internal clock mode)

The TIN input can be used as either a trigger input or a gate input when an internal clock is selected as the clock source.

Trigger input

When TIN is used as a trigger input, an active edge causes the timer to load the reload register contents and resets the internal prescaler. Following this, count operation begins. Refer to the device specific datasheet for the minimum TIN pulse width.

Figure 28-7. Trigger input operation of the 32-bit Reload Timer

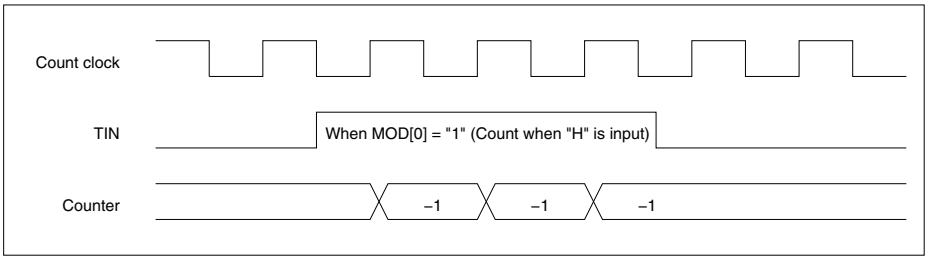


Note: T = Clock period of CLK_PERI3_PD2

Gate input

When used as a gate input, the counter only counts while the active level specified by the RLTn_TMCSR:MOD[0] bit is input to TIN. In this case, the count clock continues to operate unless it is stopped. The software trigger can be used in gate mode regardless of the gate level. Refer to the device specific datasheet for the minimum TIN pulse.

Figure 28-8. Gate input operation of the 32-bit Reload Timer



External event counter

When external event count mode is selected, TIN is used as an external event input. The counter counts on the active edge specified in RLTn_TMCSR. Refer to the device specific datasheet for the minimum TIN pulse width.

28.4 Underflow operation of the 32-bit Reload Timer

For this timer, an underflow is defined as the time when the counter value changes from 0x00000000 to 0xFFFFFFFF or when a reload occurs (RLTn_TMCSR:RELD= '1'). Therefore, an underflow occurs after (reload register setting + 1) counts.

Underflow operation of the 32-bit Reload Timer

With reference to Figure 28-9, if the RLTn_TMCSR:RELD bit is '1' and an underflow occurs, the contents of the reload register are loaded into the counter and counting continues.

If the RLTn_TMCSR:RELD bit is '0', counting stops when the counter reaches 0xFFFFFFFF.

The RLTn_TMCSR:UF bit is set when an underflow occurs. If, at the same time, the RLTn_TMCSR:INTE bit is '1', an interrupt request is generated.

Figure 28-9. Underflow operation of the 32-bit Reload Timer

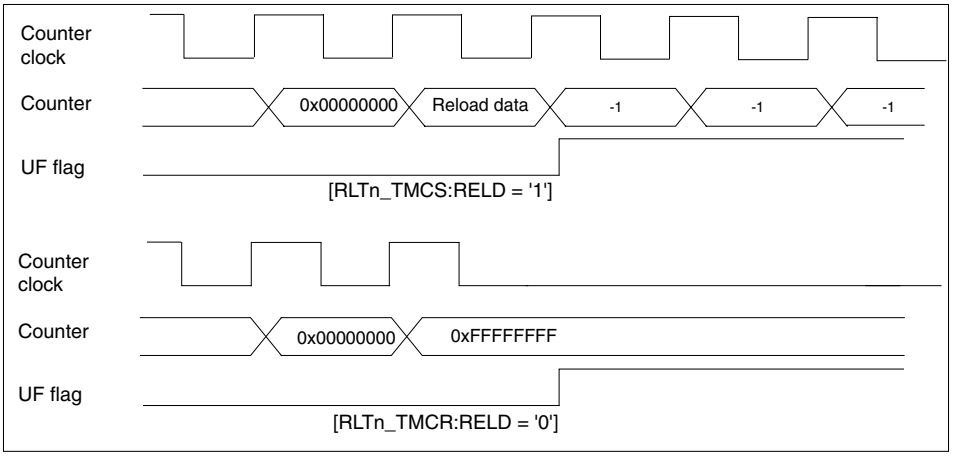
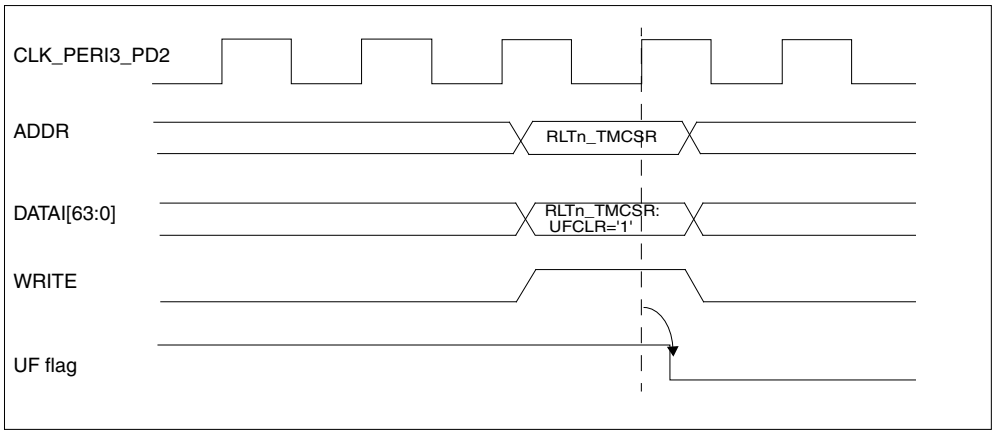


Figure 28-10 shows how the underflow flag is cleared.

Figure 28-10. Clearing the underflow flag



28.5 Output functions of the the 32-bit Reload Timer

In reload mode, the TOT output is toggled (inverts with each underflow). In one-shot mode, the TOT output is used as a pulse output that shows the configured level while counting is in progress.

Output signal functions of 32-bit Reload Timer

Figure 28-12 and Figure 28-12 show the output signal functions in reload mode and one-shot mode respectively.

The RLTn_TMCSR:OUTL bit sets the output polarity.

When RLTn_TMCSR:OUTL = '0', the initial value for the toggle output is 'L' and the one-shot pulse output is 'H' while the count is in progress.

When RLTn_TMCSR:OUTL = '1', the output waveforms are opposite.

Figure 28-11. Output signal function of 32-bit Reload Timer in reload mode

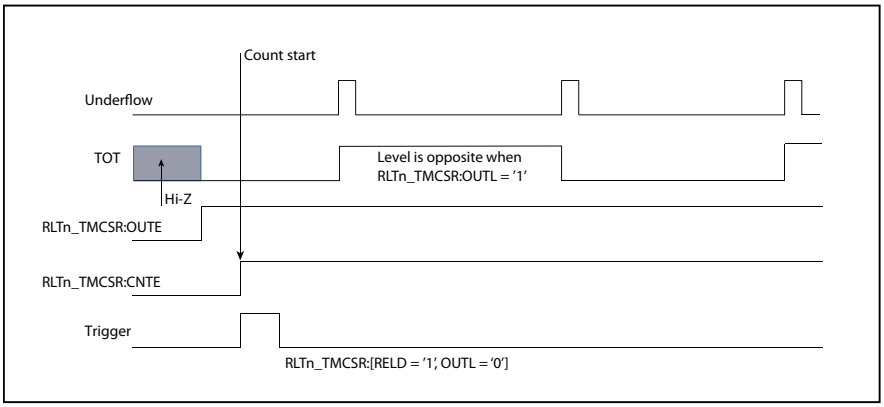
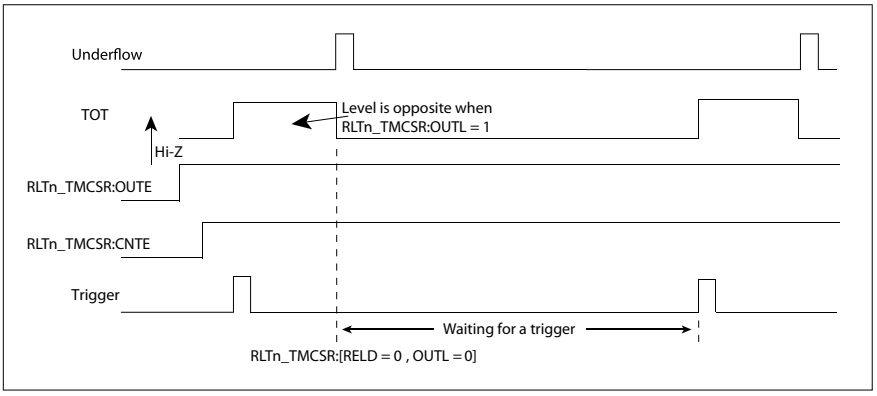


Figure 28-12. Output signal fnction of 32-bit Reload timer in one-shot mode



28.6 Counter operation state

The counter state is determined by both the RLTn_TMCSR:CNTE bit in the Timer Control Status Register and the internal WAIT signal.

Available states for the 32-bit Reload Timer are:

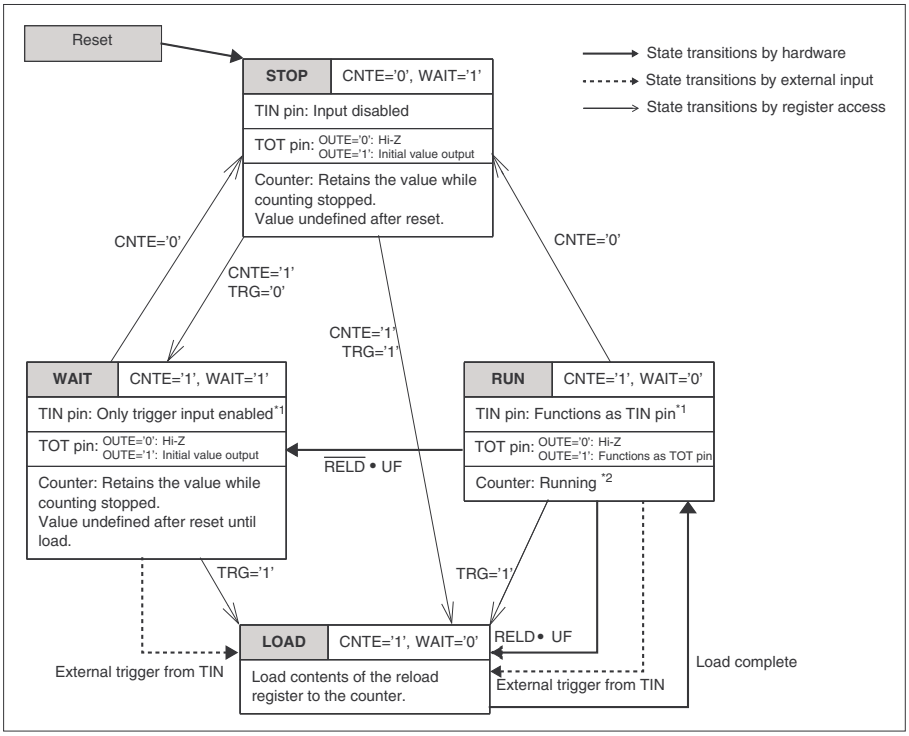
Stop state: RLTn_TMCSR:CNTE = '0' and WAIT = '1'

Wait state: RLTn_TMCSR:CNTE = '1' and WAIT = '1'

Run state: RLTn_TMCSR:CNTE = '1' and WAIT = '0'

Counter operation states

Figure 28-13. Counter operation states



*1: Before using TIN input, configure the corresponding port pin control bits correctly.

*2: In 'gate input mode': Counting can be influenced by TIN. means logic AND

The bits in the above diagram are located in the register RLTn_TMCSR.

28.7 DMA operation

DMA support is determined by the RLTn_DMACFG:ENDMAUF bit. Setting this bit enables a DMA request generation. Asserting the DMA_REQ_ACK signal acknowledges the request and the DMA_REQ signal gets deasserted.

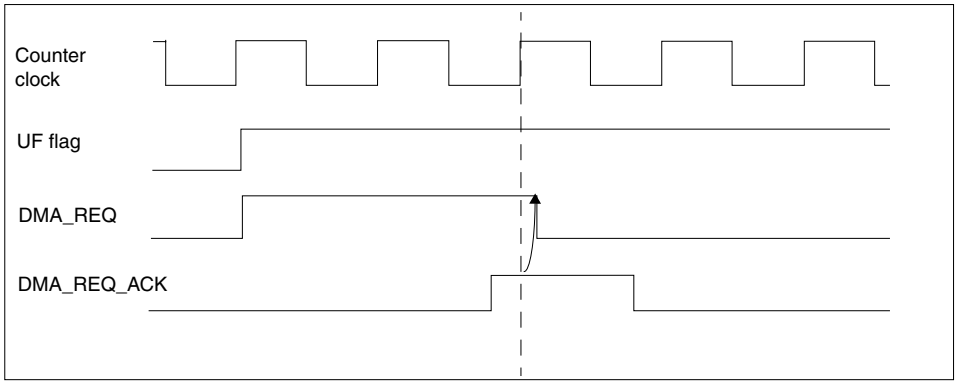
Enabling DMA support

Writing '1' to RLTn_DMACFG:ENDMAUF enables a DMA request generation when the RLTn_TMCSR:UF bit is set. However, writing '0' disables a DMA request generation even if the RLTn_TMCSR:UF bit is set.

When DMA_REQ_ACK is asserted, the DMA_REQ signal is deasserted by acknowledging the DMA request.

Figure 28-14 shows the behavior of DMA_REQ_ACK when asserted.

Figure 28-14. Deasserting DMA_REQ signal



29. Stepper Motor Controller

This chapter explains the function and operation of the Stepper Motor Controller (SMC) module.

29.1 Outline of the Stepper Motor Controller

This section describes the features and the block diagram of the Stepper Motor Controller (SMC) module. The SMC consists of an SMC Trigger Generator, Pulse Width Modulation (PWM) Pulse Generators, motor drivers and SMC output selector logic circuits.

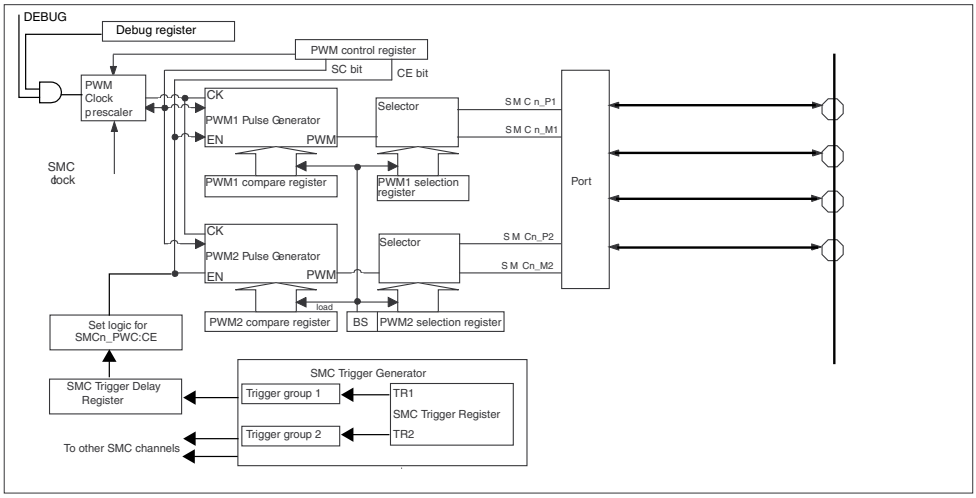
Features of Stepper Motor Controller

A Stepper Motor Controller (SMC) provides four driver outputs with high current driving capability. The two motor coils can be connected directly to the coil pins of a Stepper Motor. The motor rotation is controlled by a combination of Pulse Width Modulation (PWM) Pulse Generators and selector logic circuits. The synchronisation mechanism enables synchronous operation of the two Pulse Generators so that they operate safely. An SCM features:

- Four high-current output drivers
- Two PWM Pulse Generators with:
  - 10-bit/8-bit compare values
  - A synchronisation mechanism
  - A variable-clock prescaler
- Output selector logic (high, low, PWM, high impedance)
- Trigger delay logic for delayed start of multiple SMCs
- Support for Microcontroller Unit (MCU) debug mode

Block diagram of the Stepper Motor Controller

Figure 29-1. Block diagram of Stepper Motor Controller



29.2 Stepper Motor Controller registers

This section describes the registers of the Stepper Motor Controller.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the SMC channel. The suffix 'g' in the register name indicates that the register is an instance 'g' of the SMC group.

Registers of Stepper Motor Controller

The following registers are available for one Stepper Motor Controller:

- PWM Control Register (SMCn_PWC)
- PWM Control Set Register (SMCn_PWCS)
- PWM Control Clear Register (SMCn_PWCC)
- PWM Compare Registers (SMCn_PWC1, SMCn_PWC2)
- PWM Selection Register (SMCn_PWS)
- PWM Selection Set Register (SMCn_PWSS)
- SMC Trigger Delay Register (SMCn_PTRGDL)
- SMC Debug Register (SMCn_DEBUG)

The following registers are available for a group of six Stepper Motor Controllers:

- SMC Trigger Selection Register (SMCTGg_PTRGS)
- SMC Trigger Register (SMCTGg_PTRG)

Memory layout of Stepper Motor Controller registers

Table 29-1. Memory layout of Stepper Motor Controller registers

Offset	+1	+0
0x00000000	reserved XXXXXXXX	SMCn_PWC 00000000
0x00000002	reserved XXXXXXXX	SMCn_PWCS 00000000
0x00000004	reserved XXXXXXXX	SMCn_PWCC 00000000
0x00000006	SMCn_PWC1 000000XX XXXXXXXX	
0x00000008	SMCn_PWC2 000000XX XXXXXXXX	
0x0000000A	SMCn_PWS 00000000 00000000	
0x0000000C	SMCn_PWSS 00000000 XXXXXXXX	
0x0000000E	reserved XXXXXXXX	SMCn_PTRGDL 00000000
0x00000010	reserved XXXXXXXX	SMCn_DEBUG 00000000

Table 29-2. Memory layout of Stepper Motor Controller Trigger Group registers

Offset	+1	+0
0x00000000	SMCTGg_PTRGS 00000000 00000000	
0x00000002	reserved XXXXXXXX	SMCTGg_PTRG 00000000

29.2.1 PWM Control Register (SMCn_PWC)

The PWM Control Register (SMCn_PWC) starts and stops the SMC, configures the SMC PWM Pulse Generator to operate at 8 or 10 bits and specifies the clock input signal.

PWM Control Register (SMCn_PWC)

Figure 29-2. PWM Control Register (SMCn_PWC)

SMCn_PWC							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	P[2]	P[1]	P[0]	CE	SC	read0	read0
Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 29-3. PWM Control Register (SMCn_PWC) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6:4]	P	Clock Prescalar bits The Clock Prescaler bits set the clock prescaler value for the SMC PWM Pulse Generator as shown in Table 29-2. **Note:** Configuring the clock prescaler should be done while the counting operation is disabled (SMCn_PWC:CE = '0').
[3]	CE	Count Enable bit The Count Enable (CE) bit enables operation of the SMC module (PWM Pulse Generation and output control). '0': When the CE bit is cleared to '0' during operation of the SMC PWM Pulse Generator, the generator stops operation '1': The SMC PWM Pulse Generator starts operating. The PWM2 Pulse Generator starts one SMC clock cycle after the PWM1 Pulse Generator starts, in order to reduce the switching noise generated by the output driver This bit can be set to '1' by the SMCn_PWCS:CES bit and cleared by the SMCn_PWCC:CEC bit. This bit can also be set to '1' by using the SMC Trigger Generator. If the SMC Trigger Generator is used, this bit must first be set to '0'.

Table 29-3. PWM Control Register (SMCn_PWC) bits

Bit Position	Bit Field Name	Bit Description
[2]	SC	Switching bit The 8/10 bit Switching (SC) bit enables the SMC PWM Pulse Generator to operate with 8- or 10-bit resolution. '0': The SMC PWM Pulse Generator operates with 8-bit resolution '1': The SMC PWM Pulse Generator operates with 10-bit resolution
[1:0]	read0	-

Table 29-4. Clock Prescaler P[2:0} settings for SMC PWM Pulse Generator

P[2]	P[1]	P[0]	Clock input RCLK = (CLK_PERI0_PD2 or CLK_PERI1_PD2)	PWM cycle (at SMC clock RCLK = CLK_PERI0_PD2 or CLK_PERI1_PD2) = 80 MHz	
				SC = '0'	SC = '1'
'0'	'0'	'0'	RCLK	3.2 µs	12.8 µs
'0'	'0'	'1'	RCLK/4	12.8 µs	51.2 µs
'0'	'1'	'0'	RCLK/5	16.0 µs	64.0 µs
'0'	'1'	'1'	RCLK/6	19.2 µs	76.8 µs
'1'	'0'	'0'	RCLK/8	25.6 µs	102.4 µs
'1'	'0'	'1'	RCLK/10	32.0 µs	128.0 µs
'1'	'1'	'0'	RCLK/12	38.4 µs	153.6 µs
'1'	'1'	'1'	RCLK/16	51.2 µs	204.8 µs

Notes:

1. CLK_PERI0_PD2 is selected as an SMC clock (RCLK), for SMC in Peripheral Group 0 (PERI0).
2. CLK_PERI1_PD2 is selected as an SMC clock (RCLK), for SMC in Peripheral Group 1 (PERI1).

29.2.2 PWM Control Set Register (SMCn_PWCS)

The PWM Control Set Register (SMCn_PWCS) is used to set the value of the SMCn_PWC:CE bit.

PWM Control Set Register (SMCn_PWCS)

Figure 29-3. PWM Control Set Register (SMCn_PWCS)

SMCn_PWCS							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	CES	read0	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 29-5. PWM Control Set Register (SMCn_PWCS) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	CES	Set bit for Count Enable The Set bit for Count Enable (CES) bit sets the value of the SMCn_PWC:CE bit. Reading this bit returns '0'. '0': No effect '1': Sets the SMCn_PWC:CE bit
[2:0]	read0	-

29.2.3 PWM Control Clear Register (SMCn_PWCC)

The PWM Control Clear Register (SMCn_PWCC) is used to clear the value of the SMCn_PWC:CE bit.

PWM Control Clear Register (SMCn_PWCC)

Figure 29-4. PWM Control Clear Register (SMCn_PWCC)

SMCn_PWCC							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	CEC	read0	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 29-6. PWM Control Clear Register (SMCn_PWCC) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	CEC	Clear bit for Count Enable The Clear bit for Count Enable (CEC) bit clears the value of SMCn_PWC:CE bit. Reading this bit returns '0'. '0': No effect '1': Clears the SMCn_PWC:CE bit
[2:0]	read0	-

29.2.4 PWM Compare Registers (SMCn_PWC1, SMCn_PWC2)

The value of the two Compare Registers PWM1 and PWM2 (SMCn_PWC1, SMCn_PWC2) sets the duty cycle of the PWM pulses.

PWM1 and PWM2 Compare Registers (SMCn_PWC1, SMCn_PWC2)

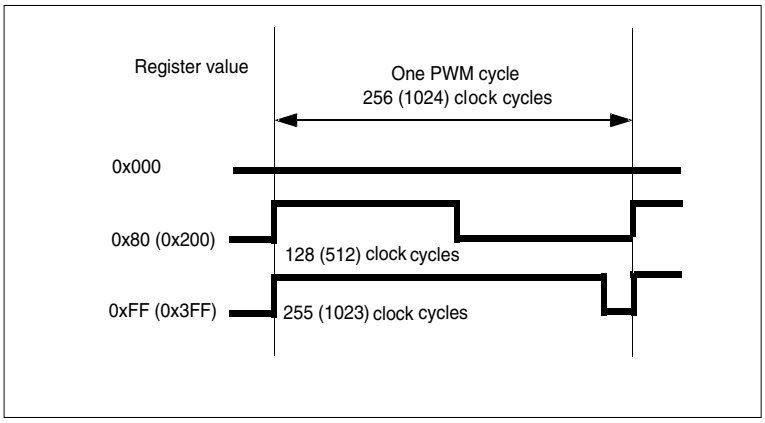
Figure 29-5. PWM1 and PWM2 Compare Registers (SMCn_PWC1, SMCn_PWC2)

SMCn_PWC2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X

Table 29-7. PWM1 and PWM2 Compare Registers (SMCn_PWC1, SMCn_PWC2) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	read0	-
[9:0]	D	PWM Compare Value The PWM compare value (D) bits are used to set the PWM duty cycle. The PWM1 and PWM2 Compare Registers can always be accessed, but the changed value is reflected in the pulse width only at the end of the current PWM cycle after '1' is set to the SMCn_PWS:BS bit. When the SMCn_PWC:SC bit is set to '0', the PWM performs an 8-bit operation and bits D[9:8] are an undefined value. The PWM1 and PWM2 Compare Registers must be written 16-bit wide. Figure 29-6 shows the relationship between compare values and generated PWM duty cycles.

Figure 29-6. Relationship between the compare register value and PWM duty cycle



29.2.5 PWM Selection Register (SMCn_PWS)

The PWM Selection Register (SMCn_PWS) sets the output state of the Stepper Motor pins (low, high, PWM, high impedance) and allows the updating of the PWM compare values.

Figure 29-7. PWM1 Selection Register (SMCn_PWS)

SMCn_PWS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	BS	P2[2]	P2[1]	P2[0]	M2[2]	M2[1]	M2[0]	read0	read0	P1[2]	P1[1]	P1[0]	M1[2]	M1[1]	M1[0]
Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-8. PWM1 Selection Register (SMCn_PWS) bits

Bit Position	Bit Field Name	Bit Description
[15]	read0	-

Table 29-8. PWM1 Selection Register (SMCn_PWS) bits

Bit Position	Bit Field Name	Bit Description
[14]	BS	Output Update The Output Update (BS) bit synchronously transfers the current configured output selection and PWM compare values to the outputs. As long as SMCn_PWS:BS is '0', changes made to the two Compare Registers (SMCn_PWC1/2) and the PWM Selection Register (SMCn_PWS) are not reflected in the output signal. This bit can also be set by the SMCn_PWSS:BSS bit. It is automatically cleared to '0' at the beginning of the next PWM cycle. '0': If the BS bit is set to '0' by the software at the same time as an automatic clear occurs, the BS bit remains set at '0'. Also the SMC PWM Pulse Generator and the selector do not load the values of the registers at the end of the current PWM cycle '1': The SMC PWM Pulse Generator and the selector load the values of the registers at the end of the current PWM cycle If the BS bit is set to '1' by the software at the same time as an automatic clear occurs, the BS bit remains set at '1' for the current PWM cycle. It is automatically cleared at the next PWM cycle. **Notes:** 1. The SMCn_PWS:BS bit has to be set to '1' to validate the configuration settings. This also applies to an initial configuration "update" when SMC operation is still inactive (SMCn_PWC:CE = '0'). 2. Though the SMCn_PWS:BS is cleared automatically at the end of a PWM cycle, it is not cleared when SMC operation is switched off inside a valid PWM cycle. Such a "pending" configuration update will be applied automatically when the SMC is switched on again (without having to reconfigure). 3. For more details on the usage of the BS bit refer to Figure 29-14.
[13:11]	P2	Plus Output 2 Selection bits The Plus Output 2 Selection bits (P2[2:0]) select the output signal for SMC2Pn. Table 29-9 shows the relationship between the output levels and the output select bits.
[10:8]	M2	Minus Output 2 Selection bits The Minus Output 2 Selection bits (M2[2:0]) select the output signal for SMC2Mn. Table 29-9 shows the relationship between the output levels and the output select bits.
[7:6]	read0	-

Table 29-8. PWM1 Selection Register (SMCn_PWS) bits

Bit Position	Bit Field Name	Bit Description
[5:3]	P1	Plus Output 1 Selection bits The Plus Output 1 Selection bits (P1[2:0]) select the output signal for SMC1Pn. Table 29-9 shows the relationship between the output levels and the output select bits.
[2:0]	M1	Minus Output 1 Selection bits The Minus Output 1 Selection bits (M1[2:0]) select the output signal for SMC1Mn. Table 29-9 shows the relationship between the output levels and the output select bits.

Table 29-9. Relationship between the output levels and the select bits

Pm[2]	Pm[1]	Pm[0]	SMCn_Pm	Mm[2]	Mm[1]	Mm[0]	SMCn_Mm
'0'	X	X	High impedance	'0'	X	X	High impedance
'1'	0	0	'L'	'1'	'0'	'0'	'L'
'1'	'0'	'1'	'H'	'1'	'0'	'1'	'H'
'1'	'1'	X	PWM pulse	'1'	'1'	X	PWM pulse

Note: m = 1 to 2 (motor coils).

The SMCn_PWS register can be read or written to using 8- and 16-bit access.

To update the SMCn_PWC1, SMCn_PWC2 and SMCn_PWS registers:

1. Update the PWM Compare Registers (SMCn_PWC1, SMCn_PWC2).
2. Configure the PWM Selection Register with SMCn_PWS:BS = '1' or set SMCn_PWS:BS via SMCn_PWSS:BSS.

29.2.6 PWM Selection Set Register (SMCn_PWSS)

The PWM Selection Set Register (SMCn_PWSS) is used to set the value of the SMCn_PWS:BS bit.

PWM Selection Set Register (SMCn_PWSS)

Figure 29-8. PWM Selection Set Register (SMCn_PWSS)

SMCn_PWSS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	BSS	read0	read0	read0	read0	read0	read0	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
Rp0	Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	-	-	-	-	-	-	-	-
0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X

Table 29-10. PWM Selection Set Register (SMCn_PWSS) bits

Bit Position	Bit Field Name	Bit Description
[15]	read0	-
[14]	BSS	Set bit for the SMCn_PWS:BS The Set bit for the SMCn_PWS:BS (BSS) sets the value of SMCn_PWS:BS bit. '0': No effect '1': Sets the SMCn_PWS:BS bit to '1' Reading this bit returns '0'.
[13:8]	read0	-
[7:0]	reserved	-

29.2.7 SMC Trigger Delay Register (SMCn_PTRGDL)

The SMC Trigger Delay Register (SMCn_PTRGDL) is used to delay the start of operation by 0 to 255 clock cycles. This delay is based on the SMC Trigger Selection Register configuration for SMCn.

SMC Trigger Delay Register (SMCn_PTRGDL)

Figure 29-9. SMC Trigger Delay Register (SMCn_PTRGDL)

SMCn_PTRGDL							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 29-11. SMC Trigger Delay Register (SMCn_PTRGDL) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	D	Trigger Delay The Trigger Delay (D) bits configure a delay of the trigger input from 0 to 255 clock cycles of the SMC clock to start the PWM operation. This trigger is generated by the SMC trigger generator. The delay of the trigger input is defined according to the following settings: '00000000': No delay '00000001' to '11111111':A delay of 1 to 255 clock cycles

29.2.8 SMC Debug Register (SMCn_DEBUG)

The SMC Debug Register (SMCn_DEBUG) is used to enable/disable the debug mode for the SMC.

SMC Debug Register (SMCn_DEBUG)

Figure 29-10. SMC Debug Register (SMCn_DEBUG)

SMCn_DEBUG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	DBGEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 29-12. SMC Debug Register (SMCn_DEBUG) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	DBGEN	Debug Mode The Debug Mode (DBGEN) bit enables/disables the debug mode for the Stepper Motor Controller. '0': Debug mode is disabled '1': Debug mode is enabled When DBGEN is set to '1' and the processor is in the debug state, PWM generation is stopped and SMCn_M1/SMCn_M2/SMCn_P1/SMCn_P2 hold their latest state. For the definition of the debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.

29.2.9 SMC Trigger Selection Register (SMCTGg_PTRGS)

The SMC Trigger Selection Register (SMCTGg_PTRGS) selects multiple SMCs that are to be triggered by the SMC Trigger Register (SMCTGg_PTRG) to start operation synchronously or with a delay. The SMCs can be combined into two trigger groups.

SMC Trigger Selection Register (SMCTGg_PTRGS)

Figure 29-11. SMC Trigger Selection Register (SMCTGg_PTRGS)

SMCTGg_PTRGS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	S25	S24	S23	S22	S21	S20	read0	read0	S15	S14	S13	S12	S11	S10
Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-13. SMC Trigger Selection Register (SMCTGg_PTRGS) bits

Bit Position	Bit Field Name	Bit Description
[15:14]	read0	-
[13]	S25	Trigger Enable for Operation of SMC<6*g+5> This bit selects SMC<6*g+5> for trigger group 2 (triggered by SMCTGg_PTRG:TR2). '0': SMC<6*g+5> is not triggered by SMCTGn_PTRG:TR2 '1': SMC<6*g+5> is triggered by SMCTGn_PTRG:TR2
[12]	S24	Trigger Enable for Operation of SMC<6*g+4> This bit selects SMC<6*g+4> for trigger group 2 (triggered by SMCTGg_PTRG:TR2). '0': SMC<6*g+4> is not triggered by SMCTGn_PTRG:TR2 '1': SMC<6*g+4> is triggered by SMCTGn_PTRG:TR2
[11]	S23	Trigger Enable for Operation of SMC<6*g+3> This bit selects SMC<6*g+3> for trigger group 2 (triggered by SMCTGg_PTRG:TR2). '0': SMC<6*g+3> is not triggered by SMCTGn_PTRG:TR2 '1': SMC<6*g+3> is triggered by SMCTGn_PTRG:TR2
[10]	S22	Trigger Enable for Operation of SMC<6*g+2> This bit selects SMC<6*g+2> for trigger group 2 (triggered by SMCTGg_PTRG:TR2). '0': SMC<6*g+2> is not triggered by SMCTGn_PTRG:TR2 '1': SMC<6*g+2> is triggered by SMCTGn_PTRG:TR2

Table 29-13. SMC Trigger Selection Register (SMCTGg_PTRGS) bits

Bit Position	Bit Field Name	Bit Description
[9]	S21	Trigger Enable for Operation of SMC<6*g+1> This bit selects SMC<6*g+1> for trigger group 2 (triggered by SMCTGg_PTRG:TR2). '0': SMC<6*g+1> is not triggered by SMCTGn_PTRG:TR2 '1': SMC<6*g+1> is triggered by SMCTGn_PTRG:TR2
[8]	S20	Trigger Enable for Operation of SMC<6*g+0> This bit selects SMC<6*g+0> for trigger group 2 (triggered by SMCTGg_PTRG:TR2). '0': SMC<6*g+0> is not triggered by SMCTGn_PTRG:TR2 '1': SMC<6*g+0> is triggered by SMCTGn_PTRG:TR2
[7:6]	read0	-
[5]	S15	Trigger Enable for Operation of SMC<6*g+5> This bit selects SMC<6*g+5> for trigger group 1 (triggered by SMCTGg_PTRG:TR1). '0': SMC<6*g+5> is not triggered by SMCTGn_PTRG:TR1 '1': SMC<6*g+5> is triggered by SMCTGn_PTRG:TR1
[4]	S14	Trigger Enable for Operation of SMC<6*g+4> This bit selects SMC<6*g+4> for trigger group 1 (triggered by SMCTGg_PTRG:TR1). '0': SMC<6*g+4> is not triggered by SMCTGn_PTRG:TR1 '1': SMC<6*g+4> is triggered by SMCTGn_PTRG:TR1
[3]	S13	Trigger Enable for Operation of SMC<6*g+3> This bit selects SMC<6*g+3> for trigger group 1 (triggered by SMCTGg_PTRG:TR1). '0': SMC<6*g+3> is not triggered by SMCTGn_PTRG:TR1 '1': SMC<6*g+3> is triggered by SMCTGn_PTRG:TR1
[2]	S12	Trigger Enable for the Operation of SMC<6*g+2> This bit selects SMC<6*g+2> for trigger group 1 (triggered by SMCTGg_PTRG:TR1). '0': SMC<6*g+2> is not triggered by SMCTGn_PTRG:TR1 '1': SMC<6*g+2> is triggered by SMCTGn_PTRG:TR1
[1]	S11	Trigger Enable for the Operation of SMC<6*g+1> This bit selects SMC<6*g+1> for trigger group 1 (triggered by SMCTGg_PTRG:TR1). '0': SMC<6*g+1> is not triggered by SMCTGn_PTRG:TR1 '1': SMC<6*g+1> is triggered by SMCTGn_PTRG:TR1

Table 29-13. SMC Trigger Selection Register (SMCTGg_PTRGS) bits

Bit Position	Bit Field Name	Bit Description
[0]	S10	Trigger Enable for the Operation of SMC<6*g+0> This bit selects SMC<6*g+0> for trigger group 1 (triggered by SMCTGg_PTRG:TR1). '0': SMC<6*g+0> is not triggered by SMCTGn_PTRG:TR1 '1': SMC<6*g+0> is triggered by SMCTGn_PTRG:TR1

Note : Each SMC must only be selected for one trigger.

29.2.10 SMC Trigger Register (SMCTGg_PTRG)

The SMC Trigger Register (SMCTGg_PTRG) is used to generate trigger signals for the two trigger groups configured in the SMC Trigger Selection Register (SMCTGg_PTRGS). The trigger group can be used to synchronously start several SMCs. The operation (count enable) starts depending on the delay configured in the SMCn_PTRGDL register.

SMC Trigger Register (SMCTGg_PTRG)

Figure 29-12. SMC Trigger Register (SMCTGg_PTRG)

SMCTGg_PTRG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	TR2	TR1
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 29-14. SMC Trigger Register (SMCTGg_PTRG) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	TR2	SMC Trigger 2 The SMC Trigger 2 bit triggers the second group of Stepper Motor Controllers. The SMCs to be triggered must be configured in SMCTGg_PTRGS:S2x. The trigger signal starts the delay counters (if enabled in SMCn_PTRGDL). If the delay is elapsed (or disabled), the count enable (SMCn_PWC:CE) bits of the selected SMCs will be set and the operation starts (PWM generation and output control). '0': No effect '1': Triggers selected SMCs via the SMCn_PTRGDL register **Note:** A second trigger should not be is applied to a group which has already been triggered. In the event that this happens, the running delay counter will be reset. This bit is automatically reset to '0' after one clock cycle.

Table 29-14. SMC Trigger Register (SMCTGg_PTRG) bits

Bit Position	Bit Field Name	Bit Description
[0]	TR1	SMC Trigger 1 The SMC Trigger 1 bit triggers the first group of Stepper Motor Controllers. The SMCs to be triggered must be configured in SMCTGg_PTRGS:S1x. The trigger signal starts the delay counters (if enabled in SMCn_PTRGDL). If the delay is elapsed (or disabled), the count enable (SMCn_PWC:CE) bits of the selected SMCs will be set and the operation starts (PWM generation and output control). '0': No effect '1': Triggers selected SMCs via the SMCn_PTRGDL register **Note:** A second trigger should not be is applied to a group which has already been triggered. In the event that this happens, the running delay counter will be reset. This bit is automatically reset to '0' after one clock cycle.

29.3 Operation of the Stepper Motor Controller

This section describes the SMC operation modes.

Setting operation of the Stepper Motor Controller

Figure 29-13. Setting of the Stepper Motor Controller

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMCn_PWC									–	P2	P1	P0	CE	SC	–	–
									X	U	U	U	U	U	X	X
SMCn_PWCS									–	–	–	–	CES	–	–	–
									X	X	X	X	1	X	X	X
SMCn_PWCC									–	–	–	–	CEC	–	–	–
									X	X	X	X	1	X	X	X
SMCn_PWC1	–	–	–	–	–	–	PWM1 H width (compare value) is set									
	X	X	X	X	X	X										
SMCn_PWC2	–	–	–	–	–	–	PWM2 H width (compare value) is set									
	X	X	X	X	X	X										
SMCn_PWS	–	BS	P2[2]	P2[1]	P2[0]	M2[2]	M2[1]	M2[0]	–	–	P1[2]	P1[1]	P1[0]	M1[2]	M1[1]	M1[0]
	X	U	U	U	U	U	U	U	X	X	U	U	U	U	U	U
SMCn_PWSS	–	BSS	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SMCn_PTRGDL									D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
									U	U	U	U	U	U	U	U
SMCTGg_PTRGS	–	–	S25	S24	S23	S22	S21	S20	–	–	S15	S14	S13	S12	S11	S10
	X	X	U	U	U	U	U	U	X	X	U	U	U	U	U	U
SMCTGg_PTRG	–	–	–	–	–	–	–	–	–	–	–	–	–	–	TR2	TR1
	X	X	X	X	X	X	X	X	X	X	X	X	X	X	U	U

U:Used bit, X:Not used bit, 1:1 is set, 0:0 is set, n:Channel no

29.3.1 Operation of the PWM Pulse Generator

Selection of motor drive signals

The motor drive signals that are output to each pin related to the SMC are selected from four types of signals for each pin by setting the PWM Selection Register (SMCn_PWS). Table 29-15 lists the selection of the motor drive signals and the PWM Selection Register settings.

Table 29-15. Motor drive signal selection and the PWM Selection Register (SMCn_PWS) setting

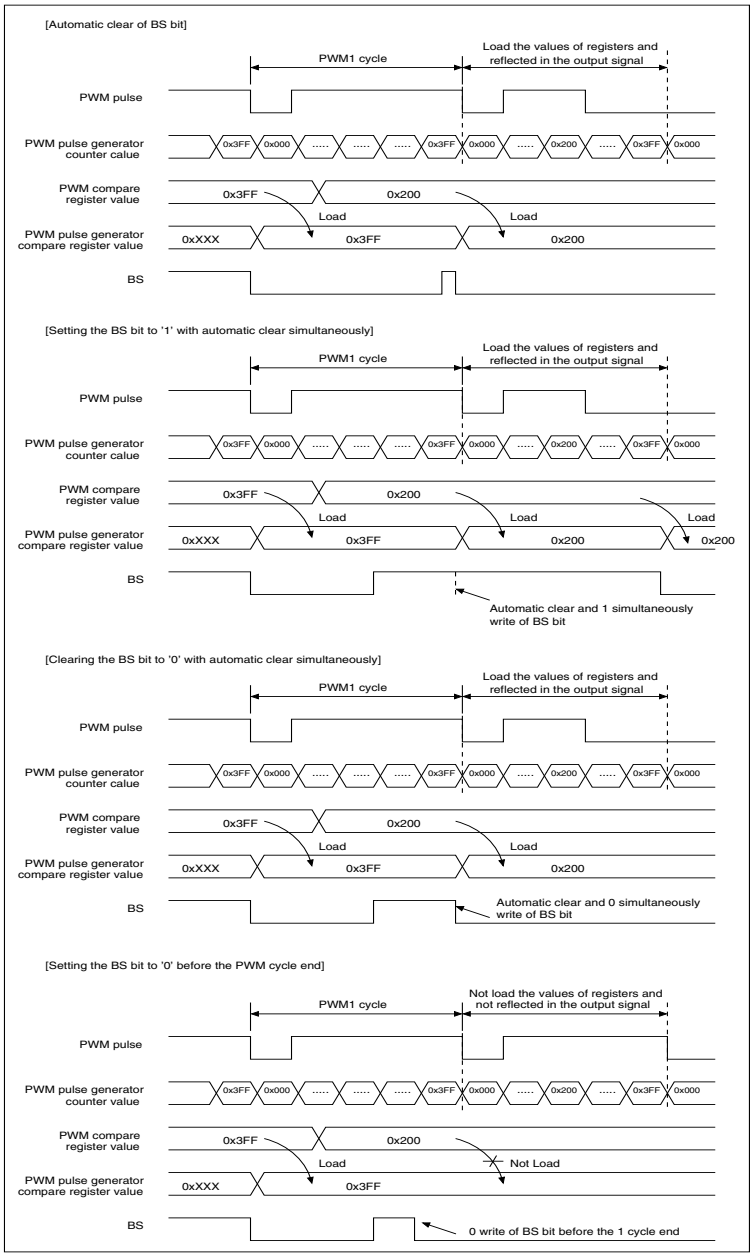
P1[2:0] bits P2[2:0] bits	SMCn_P1 output SMCn_P2 output	M1[2:0] bits M2[2:0] bits	SMCn_M1 output SMCn_M2 output
'000'	'L'	'000'	'L'
'001'	'H'	'001'	'H'
'01'X	PWM pulse	'01'X	PWM pulse
'1'XX	High impedance	'1'XX	High impedance

Update of SMC output pin, usage of the SMCn_PWS:BS bit

When SMCn_PWS is set and/or new PWM compare values are written to the PWM Compare Register(s) (i.e. SMCn_PWC1, SMCn_PWC2), '1' must be written to the SMCn_PWS:BS bit. This is to enable the transfer of both the output and PWM configuration to the SMC outputs at the end of the current PWM cycle. The SMCn_PWS:BS bit is cleared automatically at the beginning of the next PWM cycle. If '1' is written to the SMCn_PWS:BS bit and simultaneously it is cleared at the beginning of the next PWM cycle, the clearing action is cancelled (the update of the outputs will be done at the end of the next PWM cycle) and the SMCn_PWS:BS bit is set to ‘1’.

Figure 29-14 shows the timing behavior of writing PWM compare values and setting the SMCn_PWS:BS bit.

Figure 29-14. Load timing of PWM Compare Register (SMCn_PWC1, SMCn_PWC2) values

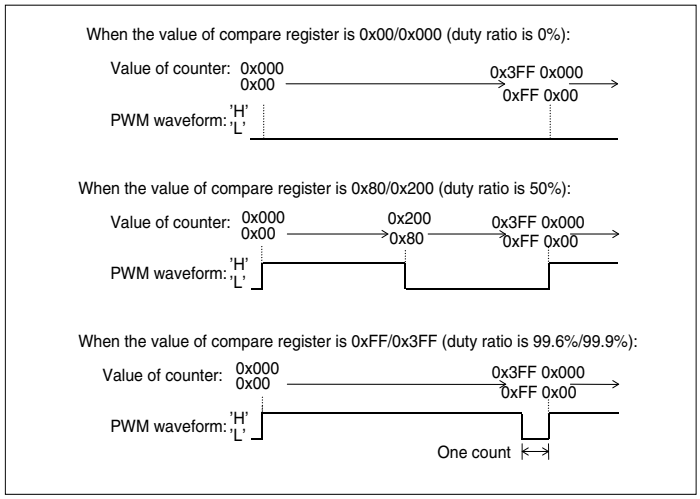


PWM Pulse Generator

When the counting operation starts (i.e. SMCn_PWC:CE = '1'), the PWM counter increments from 0x00 on the SMC clock. The PWM output pulse remains 'H' until the value of the counter matches the value set in the PWM Compare Register. It then changes to 'L' and remains so until the value of the PWM counter overflows (for 8-bit operation: 0xFF to 0x00 and for 10-bit operation: 0x3FF to 0x000).

Figure 29-15 shows the PWM waveforms generated by the SMC PWM Pulse Generator.

Figure 29-15. Examples of PWM1 and PWM2 output waveforms

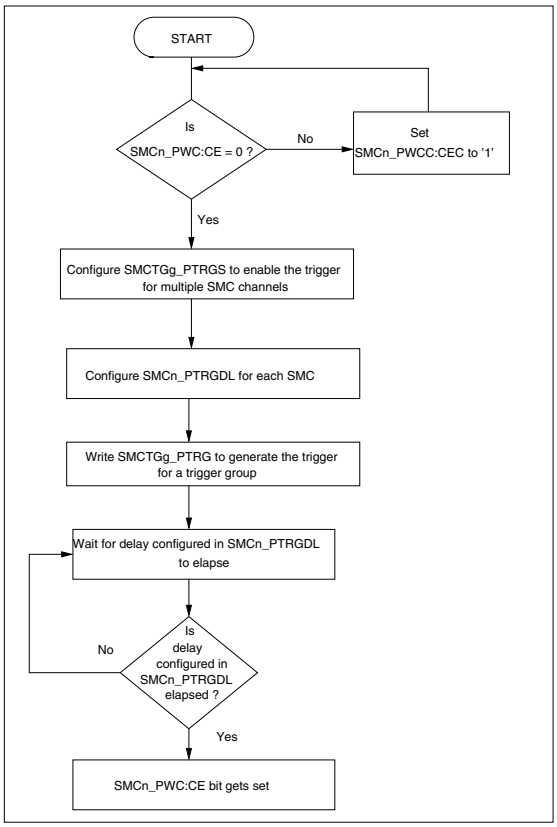


29.3.2 Operation of SMC Trigger Generator

Trigger to start the operation of one group of SMCs

The SMC Trigger Generator can generate trigger signals to start SMC operation (PWM generation and output control) for one group of SMC channels once the SMC Trigger Selection Register (SMCTGg_PTRGS), the SMC Trigger Register (SMCTGg_PTRG) and the SMC Trigger Delay Register (SMCn_PTRGDL) are configured. The SMC trigger is used to start a group of SMC channels either synchronously or with a programmed delay. Two trigger groups are supported, which can be configured independently. The SMC Trigger Select Register (SMCTGg_PTRGS) organises the SMC channels into these two groups, S1 and S2. For each Stepper Motor Controller, an individual delay can be configured in the SMC Trigger Delay Register (SMCn_PTRGDL). This delay must elapse before the count enable (i.e. SMCn_PWC:CE) bit is set by the trigger input to the selected SMCs (in the same way as the SW writes '1' to SMCn_PWCS:CES). The SMCn_PWC:CE bit is cleared by writing '1' to SMCn_PWCC:CEC to stop the SMC operation. Figure 29-16 demonstrates the program flow for the SMC Trigger Generator, and an example illustrating how the first trigger group (S1) starts SCM0 and SCM1 is shown in Figure 29-17

Figure 29-16. Flowchart for triggering multiple SMC channels



Note: The SMC Trigger Generator must only be used when the SMC channels that should be triggered are not running i.e. the count enable (SMCn_PWC:CE) bit is cleared to '0'.

Figure 29-17. Timing diagram of simultaneous triggering of SMCs

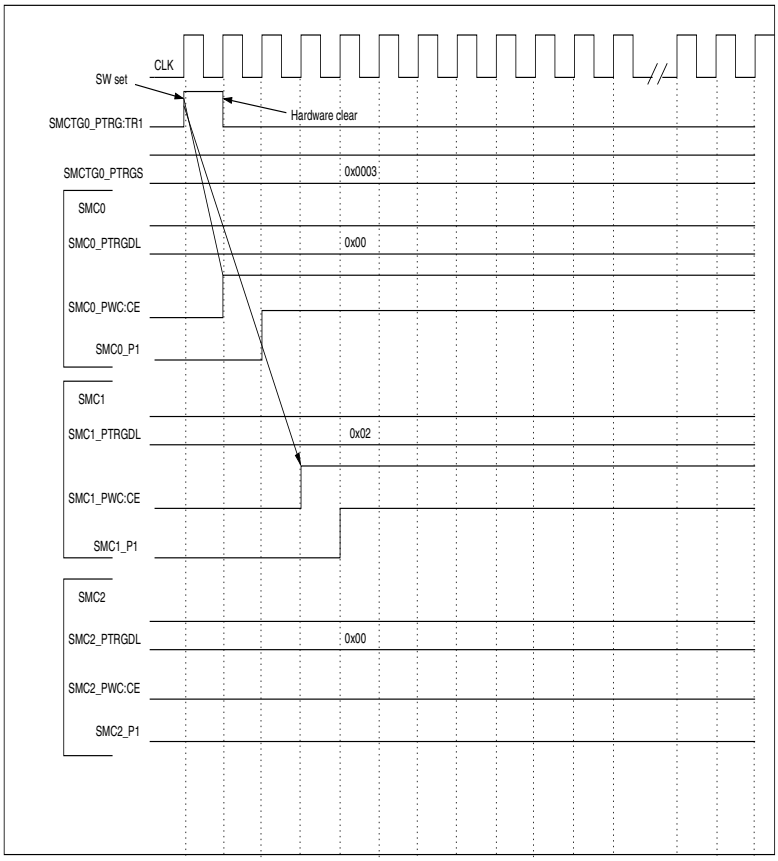


Figure 29-17 shows how trigger group S1 starts SMC0 and SMC1. SMC0 is started immediately as its SMC Trigger Delay Register setting is '0'. SMC1 is delayed relative to SMC0 by about two cycles; this delay is configured in SMC1_PTRGDL. SMC2 is not included in trigger group S1 as SMCTG0_PTRGS:S12 is '0'. The SMC0_PWC:CE and SMC1_PWC:CE bits are set by the trigger. The SMC2_PWC:CE bit is not set.

29.4 Notes on using the Stepper Motor Controller

The cautions that must be observed when changing the PWM settings and writing to the SMC Trigger Registers are described below.

Cautions when changing the PWM setting

The PWM Compare Registers 1 and 2 (SMCn_PWC1, SMCn_PWC2) and the PWM Selection Register (SMCn_PWS) can always be accessed. However, to change the setting of the PWM duty cycle or the pin output state, '1' must be written to the SMCn_PWS:BS bit after (or at the same time) a setting is written to the PWM Compare Register 1 and 2 and the PWM Selection Register. When '1' is set to the SMCn_PWS:BS bit, the new setting is enabled at the end of the current PWM cycle and the SMCn_PWS:BS bit is cleared automatically.

Also, when '1' is written to the SMCn_PWS:BS bit and simultaneously the SMCn_PWS:BS bit is reset at the end of the PWM cycle, '1' is written to the SMCn_PWS:BS and the resetting of the SMCn_PWS:BS bit is cancelled.

The BS bit can also be set by using the PWM Selection Set Register (SMCn_PWSS) to prevent any Read-Modify-Write (RMW) instruction on the PWM Selection Register.

Writing to the SMC Trigger Registers (SMCTGg_PTRG, SMCTGg_PTRGS)

To use the SMC Trigger Generator function, set the SMCn_PWC:CE bit of the SMC to be triggered to '0'. If the SMCn_PWC:CE bit is already set to '1' when asserting the group trigger signal, the SMC operation and its output is not affected.

30. CAN Controller

This chapter explains the function and operation of the CAN controller.

30.1 Overview of CAN

This section describes the features of the CAN.

Features of CAN

The CAN performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1 Mbps. For the connection to the physical layer additional transceiver hardware is required. Some of the other features of the CAN are as follows:

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1 Mbps
- 32 (up to 128) Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode (concatenation of Message Objects)
- Maskable interrupt
- Disabled automatic retransmission mode for time-triggered CAN applications
- Programmable loop-back mode for self-test operation
- Debug mode support

Note: In this chapter, the suffix 'n' in the register name denotes the CAN controller number.

30.2 CAN registers

This section describes the registers of the CAN in detail.

Registers of CAN

The following registers are available for each instance of CAN:

- CAN Device Related Register
  - CAN Output Enable Register (CANn_COER)
- CAN Protocol Related Registers
  - CAN Control Register (CANn_CTRLR)
  - Status Register (CANn_STATR)
  - Error Counter (CANn_ERRCNT)
  - Bit Timing Register (CANn_BTR)
  - Test Register (CANn_TESTR)
  - BRP Extension Register (CANn_BRPER)
- Message Interface Register Sets
  - IFx Command Request Registers (CANn_IFxCREQ)
  - IFx Command Mask Register (CANn_IFxCMSK)
  - IFx Mask Registers (CANn_IFxMSK1, CANn_IFxMSK2)
  - IFx Arbitration Registers (CANn_IF1ARB1, CANn_IF2ARB2)
  - IFx Message Control Register (CANn_IF1MCTR)
  - IFx Data A and Data B Registers (IFxDTAn, IFxDTBn)
- Message Handler Registers
  - Interrupt Register (INTRn)
  - Transmission Request Registers (TREQR1n, TREQR2n)
  - New Data Registers (NEWDT1n, NEWDT2n)
  - Interrupt Pending Registers (INTPND1n, INTPND2n)
  - Message Valid Registers (MSGVAL1n, MSGVAL2n)
  - Debug Register (DEBUGn)

Memory layout of CAN registers

The CAN module allocates an address space of 256 bytes (64 words).

The two sets of interface registers (IF1 and IF2) control the CPU access to the Message RAM. They buffer the data to be transferred to and from the RAM, avoiding conflicts between CPU accesses and message reception/transmission.

If several CAN modules are present on a device then they are located linearly in the address space with a constant offset of 256 bytes. Refer to the datasheet for the base address of each CAN module on the device.

Table 30-1. Memory layout of CAN registers

Offset	+1	+0
0x00000000	CANn_CTRLR XXXXXXXX 000X0001	
0x00000002	CANn_STATR XXXXXXXX 00000000	
0x00000004	CANn_ERRCNT 00000000 00000000	

Table 30-1. Memory layout of CAN registers

Offset	+1	+0
0x00000006	CANn_BTR X0100011 00000001	
0x00000008	CANn_INTR 00000000 00000000	
0x0000000A	CANn_TESTR XXXXXXXX 000000XX	
0x0000000C	CANn_BRPER XXXXXXXX XXXX0000	
0x0000000E	reserved XXXXXXXX XXXXXXXX	
0x00000010	CANn_IF1CREQ 0XXXXXXX 00000001	
0x00000012	CANn_IF1CMSK XXXXXXXX 00000000	
0x00000014	CANn_IF1MSK1 11111111 11111111	
0x00000016	CANn_IF1MSK2 11X11111 11111111	
0x00000018	CANn_IF1ARB1 00000000 00000000	
0x0000001A	CANn_IF1ARB2 00000000 00000000	
0x0000001C	CANn_IF1MCTR 00000000 0XXX0000	
0x0000001E	reserved XXXXXXXX XXXXXXXX	
0x00000020	CANn_IF1DTA1 00000000 00000000	
0x00000022	CANn_IF1DTA2 00000000 00000000	
0x00000024	CANn_IF1DTB1 00000000 00000000	
0x00000026	CANn_IF1DTB2 00000000 00000000	
0x00000028 - 0x0000003E	reserved XXXXXXXX XXXXXXXX	
0x00000040	CANn_IF2CREQ 0XXXXXXX 00000001	
0x00000042	CANn_IF2CMSK XXXXXXXX 00000000	
0x00000044	CANn_IF2MSK1 11111111 11111111	

Table 30-1. Memory layout of CAN registers

Offset	+1	+0
0x00000046	CANn_IF2MSK2 11X11111 11111111	
0x00000048	CANn_IF2ARB1 00000000 00000000	
0x0000004A	CANn_IF2ARB2 00000000 00000000	
0x0000004C	CANn_IF2MCTR 00000000 00000000	
0x0000004E	reserved XXXXXXXX XXXXXXXX	
0x00000050	CANn_IF2DTA1 00000000 00000000	
0x00000052	CANn_IF2DTA2 00000000 00000000	
0x00000054	CANn_IF2DTB1 00000000 00000000	
0x00000056	CANn_IF2DTB2 00000000 00000000	
0x00000058 - 0x0000007E	reserved XXXXXXXX XXXXXXXX	
0x00000080	CANn_TREQR1 00000000 00000000	
0x00000082	CANn_TREQR2 00000000 00000000	
0x00000084	CANn_TREQR3 00000000 00000000	
0x00000086	CANn_TREQR4 00000000 00000000	
0x00000088	CANn_TREQR5 00000000 00000000	
0x0000008A	CANn_TREQR6 00000000 00000000	
0x0000008C	CANn_TREQR7 00000000 00000000	
0x0000008E	CANn_TREQR8 00000000 00000000	
0x00000090	CANn_NEWDT1 00000000 00000000	
0x00000092	CANn_NEWDT2 00000000 00000000	
0x00000094	CANn_NEWDT3 00000000 00000000	

Table 30-1. Memory layout of CAN registers

Offset	+1	+0
0x00000096	CANn_NEWDT4 00000000 00000000	
0x00000098	CANn_NEWDT5 00000000 00000000	
0x0000009A	CANn_NEWDT6 00000000 00000000	
0x0000009C	CANn_NEWDT7 00000000 00000000	
0x0000009E	CANn_NEWDT8 00000000 00000000	
0x000000A0	CANn_INTPND1 00000000 00000000	
0x000000A2	CANn_INTPND2 00000000 00000000	
0x000000A4	CANn_INTPND3 00000000 00000000	
0x000000A6	CANn_INTPND4 00000000 00000000	
0x000000A8	CANn_INTPND5 00000000 00000000	
0x000000AA	CANn_INTPND6 00000000 00000000	
0x000000AC	CANn_INTPND7 00000000 00000000	
0x000000AE	CANn_INTPND8 00000000 00000000	
0x000000B0	CANn_MSGVAL1 00000000 00000000	
0x000000B2	CANn_MSGVAL2 00000000 00000000	
0x000000B4	CANn_MSGVAL3 00000000 00000000	
0x000000B6	CANn_MSGVAL4 00000000 00000000	
0x000000B8	CANn_MSGVAL5 00000000 00000000	
0x000000BA	CANn_MSGVAL6 00000000 00000000	
0x000000BC	CANn_MSGVAL7 00000000 00000000	
0x000000BE	CANn_MSGVAL8 00000000 00000000	

Table 30-1. Memory layout of CAN registers

Offset	+1	+0
0x000000C0 - 0x000000CC	reserved XXXXXXXX XXXXXXXX	
0x000000CE	reserved XXXXXXXX	CANn_COER XXXXXXX0
0x000000D0	CANn_DEBUG XXXXXXXX 00000000	

Hardware Reset Description

After a hardware reset, the CAN registers hold the values described as the initial value in Table 30-1.

Additionally, the busoff state is reset and the output CAN_TX is set to recessive (HIGH). The value 0x0001 (INIT = '1') in the CAN Control Register CANn_CTRLRL enables the software initialisation. The CAN does not influence the CAN bus until the CPU resets INIT to '0'.

The data stored in the Message RAM is not affected by a hardware reset. After power-on, the contents of the Message RAM is undefined.

30.2.1 CAN Device Related Register

30.2.1.1 CAN Output Enable Register (CANn_COER)

The CAN Output Enable Register (CANn_COER) controls whether the device pins are used as CANbus controller's TX pin or not.

CAN Output Enable Register (CANn_COER)

Figure 30-1. CAN Output Enable Register (CANn_COER)

CANn_COER							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	OE
-	-	-	-	-	-	-	RpWp
X	X	X	X	X	X	X	0

Table 30-2. CAN Output Enable Register (CANn_COER) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	reserved	
[0]	OE	Output Enable '0': CAN_TX is disabled. This pin can be used as a general purpose port or for other peripheral functions '1': CAN_TX is enabled

30.2.2 CAN Protocol Related Registers

These registers are related to the CAN protocol controller in the CAN core. They control the operating modes and the configuration of the CAN bit timing and provide status information.

30.2.2.1 CAN Control Register (CANn_CTRLR)

The CAN Control Register (CANn_CTRLR) controls the basic operation modes of the CAN controller.

CAN Control Register (CANn_CTRLR)

Figure 30-2. CAN Control Register (CANn_CTRLR)

CANn_CTRLR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	TEST	CCE	DAR	reserved	EIE	SIE	IE	INIT
-	-	-	-	-	-	-	-	RpWp	RpWp	RpWp	-	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	0	0	0	X	0	0	0	1

Table 30-3. CAN Control Register (CANn_CTRLR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7]	TEST	Test Mode Enable '0': Normal operation '1': Test mode
[6]	CCE	Configuration Change Enable '0': The CPU has no write access to the Bit Timing Register '1': The CPU has write access to the Bit Timing Register (while init = '1')
[5]	DAR	Disable Automatic Retransmission '0': Automatic retransmission of disturbed messages enabled '1': Automatic retransmission disabled
[4]	reserved	
[3]	EIE	Error Interrupt Enable '0': Disabled - No error status interrupt will be generated '1': Enabled - A change in the bits CANn_STATR:BOFF or CANn_STATR:EWARN in the status register generates an interrupt

Table 30-3. CAN Control Register (CANn_CTRLR) bits

Bit Position	Bit Field Name	Bit Description
[2]	SIE	Status Change Interrupt Enable '0': Disabled - No status change interrupt will be generated '1': Enabled - An interrupt will be generated when a message transfer is successfully completed or a CAN bus error is detected
[1]	IE	Module Interrupt Enable '0': Disabled - Module interrupt is always inactive '1': Enabled - Interrupts will be generated. The request remains active until all pending interrupts are processed
[0]	INIT	Initialization '0': Normal operation '1': Initialization is started

Note:

When the CAN is configured to work in DAR-mode (Disable Automatic Retransmission) and the host requests the transmission of several messages at the same time, only two of these messages will be transmitted. For all other requested transmit messages, the CANn_IF1MCTR:TXRQST bits will be reset, but no transmission will be started, CANn_IF1MCTR:NEWDAT and CANn_IF1MCTR:INTPND will be left unchanged. For the two messages that are transmitted, the CANn_IF1MCTR:TXRQST and CANn_IF1MCTR:NEWDAT bits will be reset and, if enabled by TXIE, INTPND will be set.

Note:

The busoff recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting INIT. If the device goes busoff, it will set INIT of its own accord, stopping all bus activities. Once INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 x 11 consecutive recessive bits) before resuming normal operation. At the end of the busoff recovery sequence, the Error Management Counters will be reset.

Note:

During the waiting time after the resetting of INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant level or continuously disturbed and to monitor the proceeding of the busoff recovery sequence.

30.2.2.2 Status Register (CANn_STATR)

The Status Register (CANn_STATR) shows the status of the CAN controller.

Status Register (CANn_STATR)

Figure 30-3. Status Register (CANn_STATR)

CANn_STATR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	BOFF	EWARN	EPASS	RXOK	TXOK	LEC[2]	LEC[1]	LEC[0]
-	-	-	-	-	-	-	-	Rp	Rp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 30-4. Status Register (CANn_STATR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7]	BOFF	Busoff Status '0': The CAN module is not in busoff state '1': The CAN module is in busoff state
[6]	EWARN	Warning Status '0': Both error counters are below the error warning limit of 96 '1': At least one of the error counters in the EML has reached the error warning limit of 96
[5]	EPASS	Error Passive '0': The CAN core is error active '1': The CAN core is in the error passive state as defined in the CAN specification
[4]	RXOK	Received a Message Successfully '0': Since this bit was last reset by the CPU, no message has been successfully received. This bit is never reset by the CAN core '1': Since this bit was last reset (to zero) by the CPU, a message has been successfully received (independent of the result of acceptance

Table 30-4. Status Register (CANn_STATR) bits

Bit Position	Bit Field Name	Bit Description
[3]	TXOK	Transmitted a Message Successfully '0': Since this bit was last reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN core '1': Since this bit was last reset by the CPU, a message has been successfully (error free and acknowledged by at least one other node) transmitted
[2:0]	LEC	Last Error Code (type of the last error to occur on the CAN bus) '000': No error '001': Stuff error. More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed '010': Form error. A fixed format part of a received frame has the wrong format '011': Ack error. The message transmitted by this CAN core was not acknowledged by another node '100': Bit1Error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant '101': Bit0Error. During the transmission of a message (or acknowledge bit or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During busoff recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the busoff recovery sequence (indicating the bus is not stuck at dominant level or continuously disturbed) '110': CRCError. The CRC checksum was incorrect in the message received; the CRC received for an incoming message does not match with the calculated CRC for the received data '111': Unused. When the LEC shows the value '111', no CAN bus event was detected since the CPU wrote this value to the LEC

The LEC field holds a code which indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. The unused code '7' may be written by the CPU to check for updates.

Status Interrupts

A Status Interrupt is generated by bits CANn_STATR:BOFF and CANn_STATR:EWARN (Error Interrupt) or by CANn_STATR:RXOK, CANn_STATR:TXOK, and CANn_STATR:LEC (Status Change Interrupt) assuming that the corresponding enable bits in the CAN Control Register are set. A change of bit CANn_STATR:EPASS or a write to CANn_STATR:RXOK, CANn_STATR:TXOK, or CANn_STATR:LEC never generates a Status Interrupt.

Reading the Status Register clears the Status Interrupt value (0x8000) in the Interrupt Register, if it is pending.

30.2.2.3 Error Counter (CANn_ERRCNT)

The Error Counter Register (CANn_ERRCNT) contains the Receive and Transmit Error Counters CANn_ERRCNTREC and CANn_ERRCNTTEC.

Error Counter (CANn_ERRCNT)

Figure 30-4. Error Counter (CANn_ERRCNT)

CANn_ERRCNT															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RP	REC[6]	REC[5]	REC[4]	REC[3]	REC[2]	REC[1]	REC[0]	TEC[7]	TEC[6]	TEC[5]	TEC[4]	TEC[3]	TEC[2]	TEC[1]	TEC[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-5. Error Counter (ERRCNTn) bits

Bit Position	Bit Field Name	Bit Description
[15]	RP	Receive Error Passive '0': The receive error counter is below the error passive level '1': The receive error counter has reached the error passive level as defined in the CAN specification
[14:8]	REC	Receive Error Counter Actual state of the receive error counter. Values between 0 and 127.
[7:0]	TEC	Transmit Error Counter Actual state of the transmit error counter. Values between 0 and 255.

30.2.2.4 Bit Timing Register (CANn_BTR)

The Bit Timing Register (CANn_BTR) enables controlling of the CAN bus controller bit timing.

Bit Timing Register (CANn_BTR)

Figure 30-5. Bit Timing Register (CANn_BTR)

CANn_BTR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	TSEG2[2]	TSEG2[1]	TSEG2[0]	TSEG1[3]	TSEG1[2]	TSEG1[1]	TSEG1[0]	SJW[1]	SJW[0]	BRP[5]	BRP[4]	BRP[3]	BRP[2]	BRP[1]	BRP[0]
-	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
X	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1

Table 30-6. Bit Timing Register (CANn_BTR) bits

Bit Position	Bit Field Name	Bit Description
[15]	reserved	
[14:12]	TSEG2	The Time Segment after the Sample Point Valid values for TSEG2 are [0 ... 7]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[11:8]	TSEG1	The Time Segment before the Sample Point Valid values for TSEG1 are [1 ... 15]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[7:6]	SJW	(Re)Synchronization Jump Width Valid programmed values are [0 ... 3]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[5:0]	BRP	Baud Rate Prescaler The value by which the peripheral clock CLK_PERI1_PD2 frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of these quanta. Valid values for the baud rate prescaler are [0 ... 63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a peripheral clock CLKP2 frequency of 8 MHz, the reset value of 0x2301 configures the CAN for a bit rate of 500 kbps. The registers are only writable if bits CCE and INIT in the CAN Control Register are set.

30.2.2.5 Test Register (CANn_TESTR)

The Test Register (CANn_TESTR) offers functionality to test the CAN bus system.

Test Register (CANn_TESTR)

Figure 30-6. Test Register (CANn_TESTR)

CANn_TESTR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	RX	TX[1]	TX[0]	LBACK	SILENT	BASIC	reserved	reserved
-	-	-	-	-	-	-	-	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	-	-
X	X	X	X	X	X	X	X	0	0	0	0	0	0	X	X

Table 30-7. Test Register (CANn_TESTR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7]	RX	Monitors the Actual Value of the CAN_RX Pin '0': The CAN bus is dominant (CAN_RX = '0') '1': The CAN bus is recessive (CAN_RX = '1')
[6:5]	TX	Control of CAN_TX Pin '00': Reset value, CAN_TX is controlled by the CAN core '01': Sample point can be monitored at CAN_TX pin '10': CAN_TX pin drives a dominant ('0') value '11': CAN_TX pin drives a recessive ('1') value
[4]	LBACK	Loop Back Mode '0': Loop back mode is disabled '1': Loop back mode is enabled
[3]	SILENT	Silent Mode '0': Normal operation '1': The module is in silent mode
[2]	BASIC	Basic Mode '0': Basic mode disabled '1': IF1 registers used as Tx buffer, IF2 registers used as Rx buffer
[1:0]	reserved	

Note: Write access to the Test Register is enabled by setting bit CANn_CTRLR:TEST in the CAN Control Register. The different test functions may be combined, but TX1-0 /= '00' disturbs message transfer.

30.2.2.6 BRP Extension Register (CANn_BRPER)

The BRP Extension Register (CANn_BRPER) contains an extension for the baud rate generator prescaler.

BRP Extension Register (CANn_BRPER)

Figure 30-7. BRP Extension Register (CANn_BRPER)

CANn_BRPER															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	BRPE[3]	BRPE[2]	BRPE[1]	BRPE[0]
-	-	-	-	-	-	-	-	-	-	-	-	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0

Table 30-8. BRP Extension Register (CANn_BRPER) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:4]	reserved	
[3:0]	BRPE	Baud Rate Prescaler Extension By programming BRPE the baud rate prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BRP (LSBs) is used.

30.2.3 Message Interface Register Sets

To avoid access conflicts between the CPU and the CAN bus controller in accessing the Message RAM, there are two sets of Interface Registers which are used to control the CPU access to the Message RAM.

Overview of Message Interface Register Sets

There are two sets of Interface Registers which are used to control the CPU access to the Message RAM. The Interface Registers avoid conflicts between CPU access to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object (see 30.3 Message Object in the Message Memory) or parts of the Message Object may be transferred between the Message RAM and the IFx Message Buffer registers in one single transfer.

The function of the two interface register sets is identical (except for test mode CANn_TESTR:BASIC). They can be used the way that one set of registers is used for data transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other. Table 30-9 gives an overview of the two Interface Register sets.

Each set of Interface Registers consists of Message Buffer Registers controlled by their own Command Registers. The Command Mask Register specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register.

Table 30-9. IF1 and IF2 Message Interface Register Sets

Address	IF1 register set	Address	IF2 register set
CAN Base + 0x10	IF1 Command Request (IF1CREQn)	CAN Base + 0x40	IF2 Command Request (IF2CREQn)
CAN Base + 0x12	IF1 Command Mask (IF1CMSKn)	CAN Base + 0x42	IF2 Command Mask (IF2CMSKn)
CAN Base + 0x14	IF1 Mask 1 (IF1MSK1n)	CAN Base + 0x44	IF2 Mask 1 (IF2MSK1n)
CAN Base + 0x16	IF1 Mask 2 (IF1MSK2n)	CAN Base + 0x46	IF2 Mask 2 (IF2MSK2n)
CAN Base + 0x18	IF1 Arbitration 1 (IF1ARB1n)	CAN Base + 0x48	IF2 Arbitration 1 (IF2ARB1n)
CAN Base + 0x1A	IF1 Arbitration 2 (IF1ARB2n)	CAN Base + 0x4A	IF2 Arbitration 2 (IF2ARB2n)
CAN Base + 0x1C	IF1 Message Control (IF1MCTRn)	CAN Base + 0x4C	IF2 Message Control (IF2MCTRn)
CAN Base + 0x20	IF1 Data A1 (IF1DTA1n)	CAN Base + 0x50	IF2 Data A1 (IF2DTA1n)
CAN Base + 0x22	IF1 Data A2 (IF1DTA2n)	CAN Base + 0x52	IF2 Data A2 (IF2DTA2n)
CAN Base + 0x24	IF1 Data B1 (IF1DTB1n)	CAN Base + 0x54	IF2 Data B1 (IF2DTB1n)
CAN Base + 0x26	IF1 Data B2 (IF1DTB2n)	CAN Base + 0x56	IF2 Data B2 (IF2DTB2n)

30.2.3.1 IFx Command Request Registers (CANn_IFxCREQ)

A message transfer is started as soon as the CPU has written the message number to the Command Request Register (CANn_IFxCREQ). With this write operation the CPU is notified that a transfer is in progress. If a CPU access to the CAN happens while the transfer is in progress then this access is delayed until the transfer has finished. After 3 to 6 CLK_PERI1_PD2 periods, the transfer between the Interface Register and the Message RAM has completed and the upcoming CPU access is executed.

IFx Command Request Registers (CANn_IFxCREQ)

Figure 30-8. IFx Command Request Registers (CANn_IFxCREQ)

CANn_IF1CREQ															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BUSY	reserved	reserved	reserved	reserved	reserved	reserved	reserved	MSGN[7]	MSGN[6]	MSGN[5]	MSGN[4]	MSGN[3]	MSGN[2]	MSGN[1]	MSGN[0]
RpWp	-	-	-	-	-	-	-	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 30-10. IFx Command Request Registers (CANn_IF1CREQ) bits

Bit Position	Bit Field Name	Bit Description
[15]	BUSY	Busy Flag '0': Reset to zero when read/write action has finished '1': Set to one when writing to the IF1 Command Request Register
[14:8]	reserved	

Table 30-10. IFx Command Request Registers (CANn_IF1CREQ) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	MSGN	Message Number 32 message buffer CAN: 0x00: Not a valid message number, interpreted as 0x20 0x01-0x20: Valid message number, the message object in the message RAM is selected for data transfer 0x21-0xFF: Not a valid message number, interpreted as 0x01-0x1F 64 message buffer CAN: 0x00: Not a valid message number, interpreted as 0x40 0x01-0x40: Valid message number, the message object in the message RAM is selected for data transfer 0x41-0xFF: Not a valid message number, interpreted as 0x01-0x3F 128 message buffer CAN: 0x00: Not a valid message number, interpreted as 0x80 0x01-0x80: Valid message number, the message object in the message RAM is selected for data transfer 0x81-0xFF: Not a valid message number, interpreted as 0x01-0x7F

Note:

The Busy Flag can only be used in CANn_TESTR:BASIC mode (see “The CAN Core provides two debug register bits (DEBUGn:DBGLB and DEBUGn:DBGSL) to supress ongoing transaction and interrupt. During debug CAN mode is changed to one of the four modes as described in Table 30-34.”). When using the Message RAM (CANn_TESTR:BASIC = '0') the hardware interface controls the access for read and write and this flag is always read as '0'.

Note:

When a Message Number that is not valid is written into the Command Request Register, the Message Number will be transformed into a valid value and that Message Object will be transferred.

30.2.3.2 IFx Command Mask Register (CANn_IFxCMSK)

The control bits of the IFx Command Mask Register (CANn_IFxCMSK) specify the transfer direction and select which of the IFx Message Buffer Registers are source or target of the data transfer.

IFx Command Mask Register (CANn_IF1CMSK)

Figure 30-9. IFx Command Mask Register (CANn_IF1CMSK)

CANn_IF1CMSK															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	WRRD	MASK	ARB	CONTROL	CIP	TXREQ	DATAA	DATAB
-	-	-	-	-	-	-	-	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 30-11. IFx Command Mask Register (CANn_IF1CMSK) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7]	WRRD	Write/Read '0': Read: Transfer data from the message object addressed by the Command Request Register into the selected Message Buffer Registers '1': Write: Transfer data from the selected Message Buffer Registers to the message object addressed by the Command Request Register The other bits of the IF1 Command Mask Register have different functions depending on the transfer direction.
[6]	MASK	Access Mask bits WRRD = '0' (read): '0': Mask bits unchanged '1': Transfer identifier mask + MDir + MXtd to IF1 Message Buffer Register WRRD = '1' (write): '0': Mask bits unchanged '1': Transfer identifier mask + MDir + MXtd to message object

Table 30-11. IFx Command Mask Register (CANn_IF1CMSK) bits

Bit Position	Bit Field Name	Bit Description
[5]	ARB	Access Arbitration bits WRRD = '0' (read): '0': Arbitration bits unchanged '1': Transfer identifier + Dir + Xtd + MsgVal to IF1 Message Buffer Register WRRD = '1' (write): '0': Arbitration bits unchanged '1': Transfer identifier + Dir + Xtd + MsgVal to message object
[4]	CONTROL	Access Control bits WRRD = '0' (read): '0': Control bits unchanged '1': Transfer control bits to IFx Message Buffer Register WRRD = '1' (Write): '0': Control bits unchanged '1': Transfer control bits to message object
[3]	CIP	Clear Interrupt Pending bit WRRD = '0' (read): '0': INTPND bit remains unchanged '1': Clear INTPND bit in the message object WRRD = '1' (write): When writing to a message object, this bit is ignored.
[2]	TXREQ	Access Transmission Request bit/Access New Data bit WRRD = '0' (read): '0': NEWDAT bit remains unchanged '1': Clear NEWDAT bit in the message object WRRD = '1' (write): '0': TXRQST bit unchanged '1': Set TXRQST bit
[1]	DATAA	Access Data Bytes 0-3 WRRD = '0' (read): '0': Data bytes 0-3 unchanged '1': Transfer data bytes 0-3 to IF1 Message Buffer Register WRRD = '1' (write): '0': Data bytes 0-3 unchanged '1': Transfer data bytes 0-3 to message object

Table 30-11. IFx Command Mask Register (CANn_IF1CMSK) bits

Bit Position	Bit Field Name	Bit Description
[0]	DATAB	Access Data Bytes 4-7 WRRD = '0' (read): '0': Data bytes 4-7 unchanged '1': Transfer data bytes 4-7 to IF1 Message Buffer Register WRRD = '1' (write): '0': Data bytes 4-7 unchanged '1': Transfer data bytes 4-7 to message object

The other bits of IFx Command Mask Register have different functions depending on the transfer direction:

Note: A read access to a Message Object can be combined with the reset of the control bits INTPND and NEWDAT. The values of these bits transferred to the IFx Message Control Register always reflect the status before resetting these bits.

30.2.3.3 IFx Mask Registers (CANn_IFxMSK1, CANn_IFxMSK2)

The bits of the Message Buffer registers mirror the Message Objects in the Message RAM. The bits of the IFx Mask Registers (CANn_IFxMSK1, CANn_IFxMSK2) specify which parts of the message arbitration registers (CANn_IFxARB1, CANn_IFx-ARB2) are evaluated.

IFx Mask Registers (CANn_IFxMSK1)

Figure 30-10. IFx Mask Registers (CANn_IFxMSK1)

CANn_IF1MSK1															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MSK[15]	MSK[14]	MSK[13]	MSK[12]	MSK[11]	MSK[10]	MSK[9]	MSK[8]	MSK[7]	MSK[6]	MSK[5]	MSK[4]	MSK[3]	MSK[2]	MSK[1]	MSK[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 30-12. IFx Mask Registers (CANn_IFxMSK1) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	MSK	Identifier Mask[15:0] For more details refer to the 30.3 Message Object in the Message Memory.

IFx Mask Registers (CANn_IFxMSK2)

Figure 30-11. IFx Mask Registers (CANn_IFxMSK2)

CANn_IF1MSK2															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MXTD	MDIR	reserved	MSK[12]	MSK[11]	MSK[10]	MSK[9]	MSK[8]	MSK[7]	MSK[6]	MSK[5]	MSK[4]	MSK[3]	MSK[2]	MSK[1]	MSK[0]
RpWp	RpWp	-	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	X	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 30-13. IFx Mask Registers (CANn_IFxMSK2) bits

Bit Position	Bit Field Name	Bit Description
[15]	MXTD	Mask Extended Identifier For more details refer to the 30.3 Message Object in the Message Memory.
[14]	MDIR	Mask Message Direction For more details refer to the 30.3 Message Object in the Message Memory.
[13]	*reserved*	
[12:0]	MSK	Identifier Mask[28:16] For more details refer to the 30.3 Message Object in the Message Memory.

30.2.3.4 IFx Arbitration Registers (CANn_IF1ARB1, CANn_IF2ARB2)

The bits of the Message Buffer registers mirror the Message Objects in the Message RAM. In case of transmission, the bits of the message arbitration registers (CANn_IF1ARB1, CAnn_IF2ARB2) specify the ID of the message. In case of reception, the bits of the message arbitration registers (CANn_IF1ARB1, CANn_IF2ARB2) specify the acceptance filter.

IFx Arbitration Registers (CANn_IF1ARB1)

Figure 30-12. IFx Arbitration Registers (CANn_IF1ARB1)

CANn_IF1ARB1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ID[15]	ID[14]	ID[13]	ID[12]	ID[11]	ID[10]	ID[9]	ID[8]	ID[7]	ID[6]	ID[5]	ID[4]	ID[3]	ID[2]	ID[1]	ID[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-14. IFx Arbitration Registers (CANn_IF1ARB1) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	ID	Message Identifier[15:0] For more details refer to the 30.3 Message Object in the Message Memory.

IFx Arbitration Registers (CANn_IF2ARB2)

Figure 30-13. IFx Arbitration Registers (CANn_IF2ARB2)

CANn_IF1ARB2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MSGVAL	XTD	DIR	ID[12]	ID[11]	ID[10]	ID[9]	ID[8]	ID[7]	ID[6]	ID[5]	ID[4]	ID[3]	ID[2]	ID[1]	ID[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-15. IFx Arbitration Registers (CANn_IF2ARB2) bits

Bit Position	Bit Field Name	Bit Description
[15]	MSGVAL	Message Valid For more details refer to the 30.3 Message Object in the Message Memory.
[14]	XTD	Extended Identifier For more details refer to the 30.3 Message Object in the Message Memory.
[13]	DIR	Message Direction For more details refer to the 30.3 Message Object in the Message Memory.
[12:0]	ID	Message Identifier[28:16] For more details refer to the 30.3 Message Object in the Message Memory.

30.2.3.5 IFx Message Control Register (CANn_IF1MCTR)

The bits of the Message Buffer registers mirror the Message Objects in the Message RAM. The bits of the Message Control Register (CANn_IF1MCTR) contain status information and control bits.

IFx Message Control Register (CANn_IF1MCTR)

Figure 30-14. IFx Message Control Register (CANn_IF1MCTR)

CANn_IF1MCTR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB	reserved	reserved	reserved	DLC[3]	DLC[2]	DLC[1]	DLC[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	-	-	-	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	X	X	X	0	0	0	0

Table 30-16. IFx Message Control Register (CANn_IF1MCTR) bits

Bit Position	Bit Field Name	Bit Description
[15]	NEWDAT	New Data For more details refer to the 30.3 Message Object in the Message Memory.
[14]	MSGLST	Message Lost For more details refer to the 30.3 Message Object in the Message Memory.
[13]	INTPND	Interrupt Pending For more details refer to the 30.3 Message Object in the Message Memory.
[12]	UMASK	Use Acceptance Mask For more details refer to the 30.3 Message Object in the Message Memory.
[11]	TXIE	Transmit Interrupt Enable For more details refer to the 30.3 Message Object in the Message Memory.
[10]	RXIE	Receive Interrupt Enable For more details refer to the 30.3 Message Object in the Message Memory.
[9]	RMTEN	Remote Enable For more details refer to the 30.3 Message Object in the Message Memory.

Table 30-16. IFx Message Control Register (CANn_IF1MCTR) bits

Bit Position	Bit Field Name	Bit Description
[8]	TXRQST	Transmit Request For more details refer to the 30.3 Message Object in the Message Memory.
[7]	EOB	End of Buffer For more details refer to the 30.3 Message Object in the Message Memory.
[6:4]	reserved	
[3:0]	DLC	Data Length Code For more details refer to the 30.3 Message Object in the Message Memory.

30.2.3.6 IFx Data A and Data B Registers (IFxDTAn, IFxDTBn)

The data bytes of CAN messages are stored in the IFx Message Buffer Registers.

IFx Data A Registers (IF1DTA1)

Figure 30-15. IFx Data A Registers (IF1DTA1)

CANn_IF1DTA1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DATA[15]	DATA[14]	DATA[13]	DATA[12]	DATA[11]	DATA[10]	DATA[9]	DATA[8]	DATA[7]	DATA[6]	DATA[5]	DATA[4]	DATA[3]	DATA[2]	DATA[1]	DATA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-17. IFx Data A Registers (IF1DTA1)

Bit Position	Bit Field Name	Bit Description
[15:0]	DATA	Data0 (Lower Byte) For more details refer to the 30.3 Message Object in the Message Memory.

IFx Data B Registers (IF1DTB1)

Figure 30-16. IFx Data B Registers (IF1DTB1)

CANn_IF1DTB1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
DATA[15]	DATA[14]	DATA[13]	DATA[12]	DATA[11]	DATA[10]	DATA[9]	DATA[8]	DATA[7]	DATA[6]	DATA[5]	DATA[4]	DATA[3]	DATA[2]	DATA[1]	DATA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-18. IFx Data B Registers (IF1DTB1)

Bit Position	Bit Field Name	Bit Description
[15:0]	DATA	Data4 (Lower Byte) For more details refer to the 30.3 Message Object in the Message Memory.

IFx Data A and Data B Registers (IFxDTAn, IFxDTBn)

In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

The data bytes of CAN messages are stored in the IFx Message Buffer Registers in the following order:

Table 30-19.

Register	Address	Address + 0 content	Address + 1 content
IFxDTA1n	x = 1: CANn base + 0x20 x = 2: CANn base + 0x50	Data(0)	Data(1)
IFxDTA2n	x = 1: CANn base + 0x22 x = 2: CANn base + 0x52	Data(2)	Data(3)
IFxDTB1n	x = 1: CANn base + 0x24 x = 2: CANn base + 0x54	Data(4)	Data(5)
IFxDTB2n	x = 1: CANn base + 0x26 x = 2: CANn base + 0x56	Data(6)	Data(7)

30.2.4 Message Handler Registers

All Message Handler registers are read-only. Their contents (TXRQST, NEWDAT, INTPND, and MSGVAL bits of each Message Object and the Interrupt Identifier) is status information provided by the Message Handler FSM.

30.2.4.1 Interrupt Register (INTRn)

The Interrupt Register (INTRn) points to the pending interrupt of highest priority.

Interrupt Register (INTRn)

Figure 30-17. Interrupt Register (INTRn)

CANn_INTR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
INTID[15]	INTID[14]	INTID[13]	INTID[12]	INTID[11]	INTID[10]	INTID[9]	INTID[8]	INTID[7]	INTID[6]	INTID[5]	INTID[4]	INTID[3]	INTID[2]	INTID[1]	INTID[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-20. Interrupt Register (INTRn) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	INTID	Interrupt Identifier The number here indicates the source of the interrupt. 32 message buffer CAN: 0x0000: No interrupt is pending 0x0001- 0x0020: Number of message object which caused the interrupt 0x0021- 0x7FFF: Unused 0x8000: Status Interrupt 0x8001-0xFFFF: Unused 64 message buffer CAN: 0x0000: No interrupt is pending 0x0001- 0x0040: Number of message object which caused the interrupt 0x0041- 0x7FFF: Unused 0x8000: Status interrupt 0x8001-0xFFFF: Unused 128 message buffer CAN: 0x0000: No interrupt is pending 0x0001- 0x0080: Number of message object which caused the interrupt 0x0081- 0x7FFF: Unused 0x8000: Status interrupt 0x8001-0xFFFF: Unused

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. If IntId is different from 0x0000 and IE is set, the interrupt line to the CPU is active. The interrupt line remains active until IntId is back to value 0x0000 (the cause of the interrupt is reset) or until IE is reset.

The Status Interrupt has the highest priority. Among the message interrupts, the Message Object's interrupt priority decreases with increasing message number.

A message interrupt is cleared by clearing the Message Object's INTPND bit. The Status Interrupt is cleared by reading the Status Register.

30.2.4.2 Transmission Request Registers (TREQR1n, TREQR2n)

The Transmission Request Registers (TREQR1n, TREQR2n) control the transmission of message objects.

Transmission Request Registers (TREQR1n)

Figure 30-18. Transmission Request Registers (TREQR1n)

CANn_TREQR1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXRQST16	TXRQST15	TXRQST14	TXRQST13	TXRQST12	TXRQST11	TXRQST10	TXRQST9	TXRQST8	TXRQST7	TXRQST6	TXRQST5	TXRQST4	TXRQST3	TXRQST2	TXRQST1
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-21. Transmission Request Registers (TREQR1n) bits

Bit Position	Bit Field Name	Bit Description
[15]	TXRQST16	Transmission Request bit '0': Message object 16 is not waiting for transmission '1': The transmission of message object 16 is requested and is not yet done
[14]	TXRQST15	Transmission Request bit
[13]	TXRQST14	Transmission Request bit
[12]	TXRQST13	Transmission Request bit
[11]	TXRQST12	Transmission Request bit
[10]	TXRQST11	Transmission Request bit
[9]	TXRQST10	Transmission Request bit
[8]	TXRQST9	Transmission Request bit
[7]	TXRQST8	Transmission Request bit
[6]	TXRQST7	Transmission Request bit
[5]	TXRQST6	Transmission Request bit
[4]	TXRQST5	Transmission Request bit
[3]	TXRQST4	Transmission Request bit
[2]	TXRQST3	Transmission Request bit
[1]	TXRQST2	Transmission Request bit '0': Message object 2 is not waiting for transmission '1': The transmission of message object 2 is requested and is not yet done

Table 30-21. Transmission Request Registers (TREQR1n) bits

Bit Position	Bit Field Name	Bit Description
[0]	TXRQST1	Transmission Request bit '0': Message object 1 is not waiting for transmission '1': The transmission of message object 1 is requested and is not yet done

Transmission Request Registers (TREQR2n)

Figure 30-19. Transmission Request Registers (TREQR2n)

CANn_TREQR2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXRQST32	TXRQST31	TXRQST30	TXRQST29	TXRQST28	TXRQST27	TXRQST26	TXRQST25	TXRQST24	TXRQST23	TXRQST22	TXRQST21	TXRQST20	TXRQST19	TXRQST18	TXRQST17
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-22. Transmission Request Registers (TREQR2n) bits

Bit Position	Bit Field Name	Bit Description
[15]	TXRQST32	Transmission Request bit '0': Message object 32 is not waiting for transmission '1': The transmission of message object 32 is requested and is not yet done
[14]	TXRQST31	Transmission Request bit
[13]	TXRQST30	Transmission Request bit
[12]	TXRQST29	Transmission Request bit
[11]	TXRQST28	Transmission Request bit
[10]	TXRQST27	Transmission Request bit
[9]	TXRQST26	Transmission Request bit
[8]	TXRQST25	Transmission Request bit
[7]	TXRQST24	Transmission Request bit
[6]	TXRQST23	Transmission Request bit
[5]	TXRQST22	Transmission Request bit
[4]	TXRQST21	Transmission Request bit
[3]	TXRQST20	Transmission Request bit
[2]	TXRQST19	Transmission Request bit

Table 30-22. Transmission Request Registers (TREQR2n) bits

Bit Position	Bit Field Name	Bit Description
[1]	TXRQST18	Transmission Request bit '0': Message object 18 is not waiting for transmission '1': The transmission of message object 18 is requested and is not yet done
[0]	TXRQST17	Transmission Request bit '0': Message object 17 is not waiting for transmission '1': The transmission of message object 17 is requested and is not yet done

These registers hold the TXRQST bits of the 32 Message Objects. By reading out the TXRQST bits, the CPU can check for which Message Object a Transmission Request is pending. The TXRQST bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or by the Message Handler after reception of a Remote Frame or after a successful transmission.

Note:

When the lowest priority message buffer is used for transmission, setting TXRQST = '0' may cause a delay of transmission, when TXRQST is set to '1' again.

Depending on the exact time when TXRQST was set to 0, the message may not be transmitted immediately after setting TXRQST = '1', but after any of the following events:

1. there is activity ongoing on the CANbus
2. a transmission request is issued on another message object
3. the CANbus is initialized by the INIT bit

In general, there is no need to cancel an ongoing transmission by setting TXRQST = 0. If the content of a message object needs to be changed while TXRQST = 1, it is sufficient to update the message object via the CPU interface registers (Identifier, DLC, Data, with TXRQST and NEWDAT, optionally TXIE). The updated content will be transmitted at the next opportunity.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 30-23. Additional flags when more than 32 message buffers exist

		addr+0	addr+1	addr+2	addr+3
TREQR 4 & 3	TXRQST 64-33 (address 0x84)	TXRQST64-57	TXRQST56-49	TXRQST48-41	TXRQST40-33
TREQR 6 & 5	TXRQST 96-65 (address 0x88)	TXRQST96-89	TXRQST88-81	TXRQST80-73	TXRQST72-65
TREQR 8 & 7	TXRQST 128-97 (address 0x8C)	TXRQST128-121	TXRQST120-113	TXRQST112-105	TXRQST104-97

30.2.4.3 New Data Registers (NEWDT1n, NEWDT2n)

The New Data Registers (NEWDT1n, NEWDT2n) indicate per message buffer that new data has been received.

New Data Registers (NEWDT1n)

Figure 30-20. New Data Registers (NEWDT1n)

CANn_NEWDT1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
NEWDAT16	NEWDAT15	NEWDAT14	NEWDAT13	NEWDAT12	NEWDAT11	NEWDAT10	NEWDAT9	NEWDAT8	NEWDAT7	NEWDAT6	NEWDAT5	NEWDAT4	NEWDAT3	NEWDAT2	NEWDAT1
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-24. New Data Registers (NEWDT1n) bits

Bit Position	Bit Field Name	Bit Description
[15]	NEWDAT16	New Data bit '0': No new data has been written into the data portion of message object 16 by the message handler since last time this flag was cleared by the CPU '1': The message handler or the CPU has written new data into the data portion of message object 16
[14]	NEWDAT15	New Data bit
[13]	NEWDAT14	New Data bit
[12]	NEWDAT13	New Data bit
[11]	NEWDAT12	New Data bit
[10]	NEWDAT11	New Data bit
[9]	NEWDAT10	New Data bit
[8]	NEWDAT9	New Data bit
[7]	NEWDAT8	New Data bit
[6]	NEWDAT7	New Data bit
[5]	NEWDAT6	New Data bit
[4]	NEWDAT5	New Data bit
[3]	NEWDAT4	New Data bit
[2]	NEWDAT3	New Data bit

Table 30-24. New Data Registers (NEWDT1n) bits

Bit Position	Bit Field Name	Bit Description
[1]	NEWDAT2	New Data bit '0': No new data has been written into the data portion of message object 2 by the message handler since last time this flag was cleared by the CPU '1': The message handler or the CPU has written new data into the data portion of message object 2
[0]	NEWDAT1	New Data bit '0': No new data has been written into the data portion of message object 1 by the message handler since last time this flag was cleared by the CPU '1': The message handler or the CPU has written new data into the data portion of message object 1

New Data Registers (NEWDT2n)

Figure 30-21. New Data Registers (NEWDT2n)

CANn_NEWDT2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
NEWDAT32	NEWDAT31	NEWDAT30	NEWDAT29	NEWDAT28	NEWDAT27	NEWDAT26	NEWDAT25	NEWDAT24	NEWDAT23	NEWDAT22	NEWDAT21	NEWDAT20	NEWDAT19	NEWDAT18	NEWDAT17
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-25. New Data Registers (NEWDT2n)

Bit Position	Bit Field Name	Bit Description
[15]	NEWDAT32	New Data bit '0': No new data has been written into the data portion of message object 32 by the message handler since last time this flag was cleared by the CPU '1': The message handler or the CPU has written new data into the data portion of message object 32
[14]	NEWDAT31	New Data bit
[13]	NEWDAT30	New Data bit
[12]	NEWDAT29	New Data bit
[11]	NEWDAT28	New Data bit

Table 30-25. New Data Registers (NEWDT2n)

Bit Position	Bit Field Name	Bit Description
[10]	NEWDAT27	New Data bit
[9]	NEWDAT26	New Data bit
[8]	NEWDAT25	New Data bit
[7]	NEWDAT24	New Data bit
[6]	NEWDAT23	New Data bit
[5]	NEWDAT22	New Data bit
[4]	NEWDAT21	New Data bit
[3]	NEWDAT20	New Data bit
[2]	NEWDAT19	New Data bit
[1]	NEWDAT18	New Data bit '0': No new data has been written into the data portion of message object 18 by the message handler since last time this flag was cleared by the CPU '1': The message handler or the CPU has written new data into the data portion of message object 18
[0]	NEWDAT17	New Data bit '0': No new data has been written into the data portion of message object 17 by the message handler since last time this flag was cleared by the CPU '1': The message handler or the CPU has written new data into the data portion of message object 17

These registers hold the NEWDAT bits of the 32 Message Objects. By reading out the NEWDAT bits, the CPU can check for which Message Object the data portion was updated. The NEWDAT bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 30-26. Additional flags when more than 32 message buffers exist

		addr+0	addr+1	addr+2	addr+3
NEWDT 4 & 3	NEWDAT 64-33 (address 0x94)	NEWDAT64-57	NEWDAT56-49	NEWDAT48-41	NEWDAT40-33
NEWDT 6 & 5	NEWDAT 96-65 (address 0x98)	NEWDAT96-89	NEWDAT88-81	NEWDAT80-73	NEWDAT72-65
NEWDT 8 & 7	NEWDAT 128-97 (address 0x9C)	NEWDAT128-121	NEWDAT120-113	NEWDAT112-105	NEWDAT104-97

30.2.4.4 Interrupt Pending Registers (INTPND1n, INTPND2n)

The Interrupt Pending Registers (INTPND1n, INTPND2n) indicate whether a message object caused an interrupt or not.

Interrupt Pending Registers (INTPND1n)

Figure 30-22. Interrupt Pending Registers (INTPND1n)

CANn_INTPND1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
INTPND16	INTPND15	INTPND14	INTPND13	INTPND12	INTPND11	INTPND10	INTPND9	INTPND8	INTPND7	INTPND6	INTPND5	INTPND4	INTPND3	INTPND2	INTPND1
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-27. Interrupt Pending Registers (INTPND1n) bits

Bit Position	Bit Field Name	Bit Description
[15]	INTPND16	Interrupt Pending bit '0': Message object 16 is not the source of an interrupt '1': Message object 16 is the source of an interrupt
[14]	INTPND15	Interrupt Pending bit
[13]	INTPND14	Interrupt Pending bit
[12]	INTPND13	Interrupt Pending bit
[11]	INTPND12	Interrupt Pending bit
[10]	INTPND11	Interrupt Pending bit
[9]	INTPND10	Interrupt Pending bit
[8]	INTPND9	Interrupt Pending bit
[7]	INTPND8	Interrupt Pending bit
[6]	INTPND7	Interrupt Pending bit
[5]	INTPND6	Interrupt Pending bit
[4]	INTPND5	Interrupt Pending bit
[3]	INTPND4	Interrupt Pending bit
[2]	INTPND3	Interrupt Pending bit
[1]	INTPND2	Interrupt Pending bit '0': Message object 2 is not the source of an interrupt '1': Message object 2 is the source of an interrupt
[0]	INTPND1	Interrupt Pending bit '0': Message object 1 is not the source of an interrupt '1': Message object 1 is the source of an interrupt

Interrupt Pending Registers (INTPND2n)

Figure 30-23. Interrupt Pending Registers (INTPND2n)

CANn_INTPND2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
INTPND32	INTPND31	INTPND30	INTPND29	INTPND28	INTPND27	INTPND26	INTPND25	INTPND24	INTPND23	INTPND22	INTPND21	INTPND20	INTPND19	INTPND18	INTPND17
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-28. Interrupt Pending Registers (INTPND2n) bits

Bit Position	Bit Field Name	Bit Description
[15]	INTPND32	Interrupt Pending bit '0': Message object 32 is not the source of an interrupt '1': Message object 32 is the source of an interrupt
[14]	INTPND31	Interrupt Pending bit
[13]	INTPND30	Interrupt Pending bit
[12]	INTPND29	Interrupt Pending bit
[11]	INTPND28	Interrupt Pending bit
[10]	INTPND27	Interrupt Pending bit
[9]	INTPND26	Interrupt Pending bit
[8]	INTPND25	Interrupt Pending bit
[7]	INTPND24	Interrupt Pending bit
[6]	INTPND23	Interrupt Pending bit
[5]	INTPND22	Interrupt Pending bit
[4]	INTPND21	Interrupt Pending bit
[3]	INTPND20	Interrupt Pending bit
[2]	INTPND19	Interrupt Pending bit
[1]	INTPND18	Interrupt Pending bit '0': Message object 18 is not the source of an interrupt '1': Message object 18 is the source of an interrupt
[0]	INTPND17	Interrupt Pending bit '0': Message object 17 is not the source of an interrupt '1': Message object 17 is the source of an interrupt

These registers hold the INTPND bits of the 32 Message Objects. By reading out the INTPND bits, the CPU can check for which Message Object an interrupt is pending. The INTPND bit of a specific Message Object can be set/reset by the CPU via

the IFx Message Interface Registers or by the Message Handler after reception or after a successful transmission of a frame. This will also affect the value of INTID in the Interrupt Register INTRn.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 30-29. Additional flags when more than 32 message buffers exist

		addr+0	addr+1	addr+2	addr+3
INTPND 4 & 3	INTPND 64-33 (address 0xA4)	INTPND64-57	INTPND56-49	INTPND48-41	INTPND40-33
INTPND 6 & 5	INTPND 96-65 (address 0xA8)	INTPND96-89	INTPND88-81	INTPND80-73	INTPND72-65
INTPND 8 & 7	INTPND 128-97 (address 0xAC)	INTPND128-121	INTPND120-113	INTPND112-105	INTPND104-97

30.2.4.5 Message Valid Registers (MSGVAL1n, MSGVAL2n)

The Message Valid Registers (MSGVAL1n, MSGVAL2n) show the validity status for each message object.

Message Valid Registers (MSGVAL1n)

Figure 30-24. Message Valid Registers (MSGVAL1n)

CANn_MSGVAL1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MSGVAL16	MSGVAL15	MSGVAL14	MSGVAL13	MSGVAL12	MSGVAL11	MSGVAL10	MSGVAL9	MSGVAL8	MSGVAL7	MSGVAL6	MSGVAL5	MSGVAL4	MSGVAL3	MSGVAL2	MSGVAL1
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-30. Message Valid Registers (MSGVAL1n) bits

Bit Position	Bit Field Name	Bit Description
[15]	MSGVAL16	Message Valid bit '0': Message object 16 is ignored by the message handler '1': Message object 16 is configured and should be considered by the message handler
[14]	MSGVAL15	Message Valid bit
[13]	MSGVAL14	Message Valid bit
[12]	MSGVAL13	Message Valid bit
[11]	MSGVAL12	Message Valid bit
[10]	MSGVAL11	Message Valid bit
[9]	MSGVAL10	Message Valid bit
[8]	MSGVAL9	Message Valid bit
[7]	MSGVAL8	Message Valid bit
[6]	MSGVAL7	Message Valid bit
[5]	MSGVAL6	Message Valid bit
[4]	MSGVAL5	Message Valid bit
[3]	MSGVAL4	Message Valid bit
[2]	MSGVAL3	Message Valid bit
[1]	MSGVAL2	Message Valid bit '0': Message object 2 is ignored by the message handler '1': Message object 2 is configured and should be considered by the message handler

Table 30-30. Message Valid Registers (MSGVAL1n) bits

Bit Position	Bit Field Name	Bit Description
[0]	MSGVAL1	Message Valid bit '0': Message object 1 is ignored by the message handler '1': Message object 1 is configured and should be considered by the message handler

Message Valid Registers (MSGVAL2n)

Figure 30-25. Message Valid Registers (MSGVAL2n)

CANn_MSGVAL2															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MSGVAL32	MSGVAL31	MSGVAL30	MSGVAL29	MSGVAL28	MSGVAL27	MSGVAL26	MSGVAL25	MSGVAL24	MSGVAL23	MSGVAL22	MSGVAL21	MSGVAL20	MSGVAL19	MSGVAL18	MSGVAL17
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-31. Message Valid Registers (MSGVAL2n) bits

Bit Position	Bit Field Name	Bit Description
[15]	MSGVAL32	Message Valid bit '0': Message object 32 is ignored by the message handler '1': Message object 32 is configured and should be considered by the message handler
[14]	MSGVAL31	Message Valid bit
[13]	MSGVAL30	Message Valid bit
[12]	MSGVAL29	Message Valid bit
[11]	MSGVAL28	Message Valid bit
[10]	MSGVAL27	Message Valid bit
[9]	MSGVAL26	Message Valid bit
[8]	MSGVAL25	Message Valid bit
[7]	MSGVAL24	Message Valid bit
[6]	MSGVAL23	Message Valid bit
[5]	MSGVAL22	Message Valid bit
[4]	MSGVAL21	Message Valid bit
[3]	MSGVAL20	Message Valid bit
[2]	MSGVAL19	Message Valid bit

Table 30-31. Message Valid Registers (MSGVAL2n) bits

Bit Position	Bit Field Name	Bit Description
[1]	MSGVAL18	Message Valid bit '0': Message object 18 is ignored by the message handler '1': Message object 18 is configured and should be considered by the message handler
[0]	MSGVAL17	Message Valid bit '0': Message object 17 is ignored by the message handler '1': Message object 17 is configured and should be considered by the message handler

These registers hold the MSGVAL bits of the 32 Message Objects. By reading out the MSGVAL bits, the CPU can check which Message Object is valid. The MSGVAL bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 30-32. Additional flags when more than 32 message buffers exist

		addr+0	addr+1	addr+2	addr+3
MSGVAL 4 & 3	MSGVAL 64-33 (address 0xB4)	MSGVAL64-57	MSGVAL56-49	MSGVAL48-41	MSGVAL40-33
MSGVAL 6 & 5	MSGVAL 96-65 (address 0xB8)	MSGVAL96-89	MSGVAL88-81	MSGVAL80-73	MSGVAL72-65
MSGVAL 8 & 7	MSGVAL 128-97 (address 0xBC)	MSGVAL128-121	MSGVAL120-113	MSGVAL112-105	MSGVAL104-97

30.2.5 Debug Register (DEBUGn)

This register is related to the Debug mode feature of CAN Controller. The Debug register is to enable debug mode (i.e. User or Silent or Loop Back or Loopback combined with Silent Mode Functionality) of CAN, when CPU generates DEBUG signal.

Debug Register (DEBUGn)

Figure 30-26. Debug Register (DEBUGn)

CANn_DEBUG															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	DBGLB	DBGSL
-	-	-	-	-	-	-	-	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 30-33. Debug Register (DEBUGn) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:2]	read0	-
[1]	DBGLB	Debug Loop Back Mode Enable 0: Disable internal loop back mode in case of debug mode 1: Enable internal loop back mode in case of debug mode
[0]	DBGSL	Debug Silent Mode Enable 0: Disable internal silent mode in case of debug mode 1: Enable internal silent mode in case of debug mode

30.3 Message Object in the Message Memory

There are 32 Message Objects (up to 128 depending on the implementation) in the Message RAM. To avoid conflicts between CPU access to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled via the IFx Interface Registers.

Structure of a Message Object in the Message Memory

Figure 30-27 gives an overview of the two structure of a Message Object

Figure 30-27. Structure of a Message Object in the Message Memory

Message Object												
UMASK	MSK28-0	MXTD	MDIR	EOB	NEWDAT		MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST
MSGVAL	ID28-0	XTD	DIR	DLC3-0	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

MSGVAL Message Valid

0 The Message Object is ignored by the Message Handler.

1 The Message Object is configured and should be considered by the Message Handler.

Note: The CPU must reset the MSGVAL bit of all unused Messages Objects during the initialization before it resets bit INIT in the CAN Control Register. This bit must also be reset before the identifier ID28-0, the control bits XTD, DIR, or the Data Length Code DLC3-0 are modified, or if the Message Object is no longer required.

UMASK Use Acceptance Mask

0 Mask ignored.

1 Use Mask (MSK28-0, MXTD, and MDIR) for acceptance filtering.

Note: If the UMASK bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MSGVAL is set to "1".

ID28-0 Message Identifier

ID28 - ID0 29-bit Identifier ("Extended Frame").

ID28 - ID18 11-bit Identifier ("Standard Frame").

MSK28-0 Identifier Mask

0 The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.

1 The corresponding identifier bit is used for acceptance filtering.

XTD Extended Identifier

0 The 11-bit ("standard") Identifier will be used for this Message Object.

1 The 29-bit ("extended") Identifier will be used for this Message Object.

MXTD Mask Extended Identifier

0 The extended identifier bit (XTD) has no effect on the acceptance filtering

1 The extended identifier bit (XTD) is used for acceptance filtering.

Note: When 11-bit ("standard") Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits MSK28 to MSK18 are considered.

DIR Message Direction

0 Direction = receive: On TXRQST, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object.

1 Direction = transmit: On TXRQST, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TXRQST bit of this Message Object is set (if RMTEN = one).

MDIR Mask Message Direction

0 The message direction bit (DIR) has no effect on the acceptance filtering.

1 The message direction bit (DIR) is used for acceptance filtering. The Arbitration Registers ID28-0, XTD, and DIR are used to define the identifier and type of outgoing messages and are used (together with the mask registers MSK28-0, MXTD and MDIR) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction = receive (Data Frame) or Direction = transmit(Remote Frame). Extended frames can be stored only in Message Objects with XTD = one, standard frames in Message Objects with XTD = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number. For details see Acceptance Filtering of Received Messages.

EOB End of Buffer

0 Message Object belongs to a FIFO Buffer and is not the last Message Object of that FIFO Buffer.

1 Single Message Object or last Message Object of a FIFO Buffer.

Note: This bit is used to concatenate two ore more Message Objects (up to 32) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer) this bit must always be set to 1. For details on the concatenation of Message Objects see Configuration of a FIFO Buffer.

NEWDAT New Data

0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU.

1 The Message Handler or the CPU has written new data into the data portion of this Message Object.

MSGLST Message Lost (only valid for Message Objects with direction = receive)

0 No message lost since last time this bit was reset by the CPU.

1 The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message.

RXIE Receive Interrupt Enable

0 INTPND will be left unchanged after a successful reception of a frame.

1 INTPND will be set after a successful reception of a frame.

TXIE Transmit Interrupt Enable

0 INTPND will be left unchanged after the successful transmission of a frame.

1 INTPND will be set after a successful transmission of a frame.

INTPND Interrupt Pending

0 This message object is not the source of an interrupt.

1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.

RMTEN Remote Enable

0 At the reception of a Remote Frame, TXRQST is left unchanged.

1 At the reception of a Remote Frame, TXRQST is set.

TXRQST Transmit Request

0	This Message Object is not waiting for transmission.
1	The transmission of this Message Object is requested and is not yet done.

Note:

When the lowest priority message buffer is used for transmission, setting TXRQST = '0' may cause a delay of transmission, when TXRQST is set to '1' again.

Depending on the exact time when TXRQST was set to 0, the message may not be transmitted immediately after setting TXRQST = '1', but after any of the following events:

1. There is activity ongoing on the CANbus
2. A transmission request is issued on another message object
3. The CANbus is initialized by the INIT bit

In general, there is no need to cancel an ongoing transmission by setting TXRQST = 0. If the content of a message object needs to be changed while TXRQST = 1, it is sufficient to update the message object via the CPU interface registers (Identifier, DLC, Data, with TXRQST and NEWDAT, optionally TXIE). The updated content will be transmitted at the next opportunity.

DLC3-0	Data Length Code
0-8	Data Frame has 0-8 data bytes.
9-15	Data Frame has 8 data bytes.

Note: The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.

Data0	1st data byte of a CAN Data Frame
Data1	2nd data byte of a CAN Data Frame
Data2	3rd data byte of a CAN Data Frame
Data3	4th data byte of a CAN Data Frame
Data4	5th data byte of a CAN Data Frame
Data5	6th data byte of a CAN Data Frame
Data6	7th data byte of a CAN Data Frame
Data7	8th data byte of a CAN Data Frame

Note:

Byte Data0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data7 is the last. When the Message Handler stores a Data Frame, it will write all the eight data bytes into a Message Object. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by non specified values.

30.4 Functional description

This section provides an overview of the CAN module's operating modes and how to use them.

Software Initialisation

The software initialization is started by setting the bit INIT in the CAN Control Register CTRLRLn, either by software or by a hardware reset, or by going Bus_Off.

While INIT is set, all message transfer from and to the CAN bus is stopped, the status of the CAN bus output CAN_TX is recessive (HIGH). The counters of the EML are unchanged. Setting INIT does not change any configuration register.

To initialize the CAN Controller, the CPU has to set up the Bit Timing Register BTRn and each Message Object. If a Message Object is not needed, it is sufficient to set it's MSGVAL bit to not valid. Otherwise, the whole Message Object has to be initialized.

Access to the Bit Timing Register BTRn and to the BRP Extension Register BRPERn for the configuration of the bit timing is enabled when both bits INIT and CCE in the CAN Control Register CTRLRLn are set.

Resetting INIT (by CPU only) finishes the software initialisation. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and starts the message transfer.

The initialization of the Message Objects is independent of INIT and can be done on the fly, but the Message Objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer.

To change the configuration of a Message Object during normal operation, the CPU has to start by setting MSGVAL bit to not valid. When the configuration is completed, MSGVAL bit is set to valid again.

CAN Message Transfer

Once the CAN is initialized and INIT is reset to zero, the CAN's CAN Core synchronizes itself to the CAN bus and starts the message transfer.

Received messages are stored into their appropriate Message Objects if they pass the Message Handler's acceptance filtering. The whole message including all arbitration bits, DLC and eight data bytes is stored into the Message Object. If the Identifier Mask is used, the arbitration bits, which are masked to "don't care", may be overwritten in the Message Object.

The CPU may read or write each message any time via the Interface Registers, the Message Handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted are updated by the CPU. If a permanent Message Object (arbitration and control bits set up during configuration) exists for the message, only the data bytes are updated and then TXRQST bit with NEWDAT bit are set to start the transmission. If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.

The transmission of any number of Message Objects may be requested at the same time, they are transmitted subsequently according to their internal priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started. Depending on the configuration of the Message Object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. By default, this means that automatic retransmission is enabled. It can be disabled to enable the CAN to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment. The Disabled Automatic Retransmission mode is enabled by programming bit DAR in the CAN Control Register CTRLRn to '1'. In this operation mode the programmer has to consider the different behaviour of bits TXRQST and NEWDAT in the Control Registers of the Message Buffers:

When a transmission starts bit TXRQST of the respective Message Buffer is reset, while bit NEWDAT remains set.

When the transmission completed successfully bit NEWDAT is reset.

When a transmission failed (lost arbitration or error) bit NEWDAT remains set. To restart the transmission the CPU has to set TXRQST back to one.

When the host requests the transmission of several messages at the same time, only two of these messages will be transmitted. For all other requested transmit messages, the TXRQST bits will be reset, but no transmission will be started, NEWDAT and INTPND will be left unchanged. For the two messages that are transmitted, the TXRQST and NEWDAT bits will be reset and, if enabled by TXIE, INTPND will be set.

Test Mode

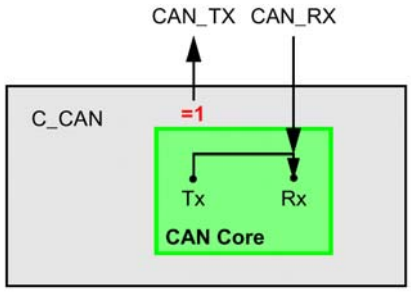
The Test Mode is entered by setting bit TEST in the CAN Control Register to '1'. In Test Mode the bits TX1, TX0, LBACK, SILENT and BASIC in the Test Register TESTRn are writable. Bit RX monitors the state of pin CAN_RX and therefore is only readable. All Test Register functions are disabled when bit TEST is reset to zero.

Silent Mode

The CAN Core can be set in Silent Mode by programming the Test Register TESTRn bit SILENT to '1'. In Silent Mode, the CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Core is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analyse the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). Figure 30-28 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

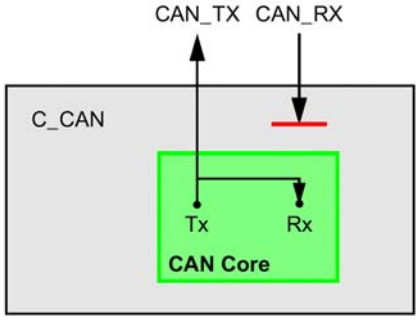
Figure 30-28. CAN Core in Silent Mode



Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register TESTRn bit LBACK to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into a Receive Buffer. Figure 30-29 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Loop Back Mode.

Figure 30-29. CAN Core in Loop Back Mode

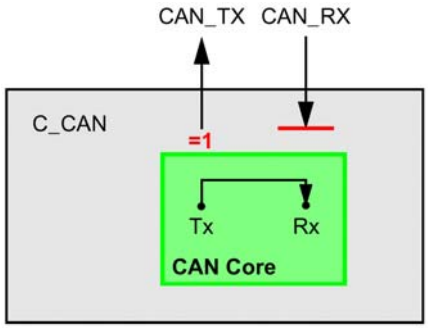


This mode is provided for self-test functions. To be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the CAN Core performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN_RX input pin is disregarded by the CAN Core. The transmitted messages can be monitored at the CAN_TX pin.

Loop Back combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBACK and SILENT to '1' at the same time. This mode can be used for a 'Hot Selftest', meaning the CAN can be tested without affecting a running CAN system connected to the pins CAN_TX and CAN_RX. In this mode the CAN_RX pin is disconnected from the CAN Core and the CAN_TX pin is held recessive. Figure 30-30 CAN Core in Loop Back combined with Silent Mode shows the connection of signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

Figure 30-30. CAN Core in Loop Back combined with Silent Mode



Debug Mode

The CAN Core provides two debug register bits (DEBUGn:DBGLB and DEBUGn:DBGSL) to supress ongoing transaction and interrupt. During debug CAN mode is changed to one of the four modes as described in Table 30-34.

Table 30-34. CAN Mode using Debug Mode

DBGLB	DBGSL	Behaviour at Break
0	0	User Mode
0	1	Silent Mode
1	0	Loop Back Mode
1	1	Loop Back combined with Silent Mode

These User/Silent/Loop Back/Loop Back combined with Silent modes are functionally similar to their respective CAN modes.

Note: When CAN comes out of break(Debug Mode) then there is possibility of transaction error.

Basic Mode

The CAN Core can be set in Basic Mode by programming the Test Register TESTRn bit BASIC to '1'. In this mode the CAN module runs without the Message RAM.

The IF1 Registers are used as Transmit Buffer. The transmission of the contents of the IF1 Registers is requested by writing the BUSY bit of the IF1 Command Request Register IF1CREQn to '1'. The IF1 Registers are locked while the BUSY bit is set. The BUSY bit indicates that the transmission is pending.

As soon the CAN bus is idle, the IF1 Registers are loaded into the shift register of the CAN Core and the transmission is started. When the transmission has completed, the BUSY bit is reset and the locked IF1 Registers are released.

A pending transmission can be aborted at any time by resetting the BUSY bit in the IF1 Command Request Register IF1CREQn while the IF1 Registers are locked. If the CPU has reset the BUSY bit, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The IF2 Registers are used as Receive Buffer. After the reception of a message the contents of the shift register is stored into the IF2 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing the BUSY bit of the IF2 Command Request Register IF1CREQn to '1', the contents of the shift register are stored into the IF2 Registers.

In Basic Mode the evaluation of all Message Object related control and status bits and of the control bits of the IFx Command Mask Registers IF1CMSKn is turned off. The message number of the Command Request Registers is not evaluated. The NEWDAT and MSGLST bits of the IF2 Message Control Register IF2MCTRn retain their function, DLC3-0 will show the received DLC, the other control bits will be read as '0'.

Software control of Pin CAN_TX

Four output functions are available for the CAN transmit pin CAN_TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor CAN_Core's bit timing and it can drive constant dominant or recessive values. The last two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the CAN bus' physical layer. The output mode of pin CAN_TX is selected by programming the Test Register TESTRn bits TX1 and TX0 as described in 30.2.2 CAN Protocol Related Registers.

The three test functions for pin CAN_TX interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes Loop Back Mode, Silent Mode, or Basic Mode are selected.

30.5 CAN application

This section describes how to use the CAN module in the application

Management of Message Objects

The configuration of the Message Objects in the Message RAM will (with the exception of the bits MSGVAL, NEWDAT, INTPND, and TXRQST) not be affected by resetting the chip. All the Message Objects must be initialized by the CPU or they must be not valid (MSGVAL = '0') and the bit timing must be configured before the CPU clears the INIT bit in the CAN Control Register.

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data field of one of the two interface register sets to the desired values. By writing to the corresponding IFx Command Request Register, the IFx Message Buffer Registers are loaded into the addressed Message Object in the Message RAM.

When the INIT bit in the CAN Control Register is cleared, the CAN Protocol Controller state machine of the CAN_Core and the Message Handler State Machine control the CAN's internal data flow. Received messages that pass the acceptance filtering are stored into the Message RAM, messages with pending transmission request are loaded into the CAN_Core's Shift Register and are transmitted via the CAN bus.

The CPU reads received messages and updates messages to be transmitted via the IFx Interface Registers. Depending on the configuration, the CPU is interrupted on certain CAN message and CAN error events.

Message Handler State Machine

The Message Handler controls the data transfer between the Rx/Tx Shift Register of the CAN Core, the Message RAM and the IFx Registers.

The Message Handler FSM controls the following functions:

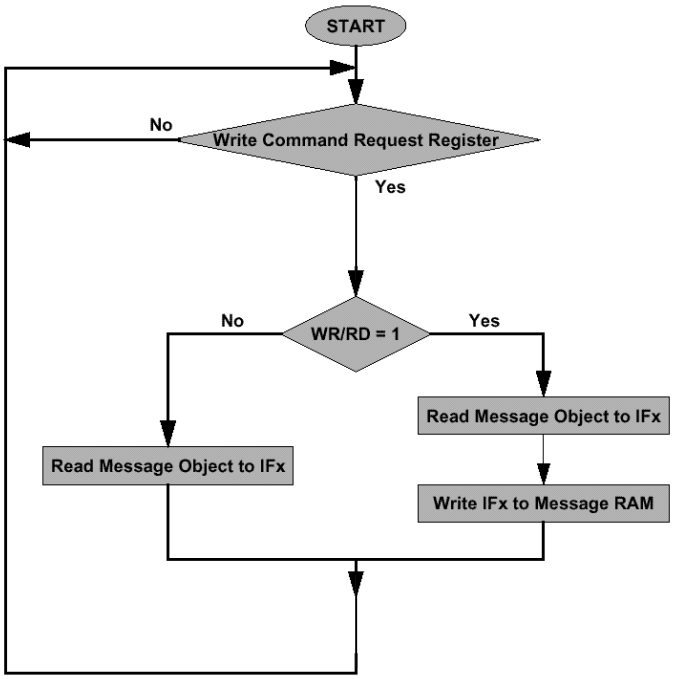
- Data Transfer from IFx Registers to the Message RAM
- Data Transfer from Message RAM to the IFx Registers
- Data Transfer from Shift Register to the Message RAM
- Data Transfer from Message RAM to Shift Register
- Data Transfer from Shift Register to the Acceptance Filtering unit
- Scanning of Message RAM for a matching Message Object
- Handling of TXRQST flags.
- Handling of interrupts.

Data Transfer from/to Message RAM

When the CPU initiates a data transfer between the IFx Registers and Message RAM, the Message Handler sets an internal busy signal which delays a consecutive access. After the transfer has completed, the busy signal is set back and the consecutive access is executed.

The respective Command Mask Register specifies whether a complete Message Object or only parts of it will be transferred. Due to the structure of the Message RAM it is not possible to write single bits/bytes of one Message Object, it is always necessary to write a complete Message Object into the Message RAM. Therefore the data transfer from the IFx Registers to the Message RAM requires a read-modify-write cycle. First the parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer Registers are transferred to the Message Object.

Figure 30-31. Data Transfer between IFx Registers and Message RAM



After the partial write of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be set to the actual contents of the selected Message Object.

After the partial read of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be left unchanged.

Transmission of Messages

If the shift register of the CAN Core cell is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the MSGVAL bits in the Message Valid Register and the TXRQST bits in the Transmission Request Register are evaluated. The valid Message Object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The Message Object's NEWDAT bit is reset.

After a successful transmission and if no new data was written to the Message Object (NEWDAT = '0') since the start of the transmission, the TXRQST bit will be reset. If TXIE is set, INTPND will be set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

Acceptance Filtering of Received Messages

When the arbitration and control field (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the Rx/Tx Shift Register of the CAN Core, the Message Handler FSM starts the scanning of the Message RAM for a matching valid Message Object.

To scan the Message RAM for a matching Message Object, the Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register. Then the arbitration and mask fields (including MSGVAL, UMASK, NEWDAT, and EOB) of Message Object 1 are loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached.

If a match occurs, the scanning is stopped and the Message Handler FSM proceeds depending on the type of frame (Data Frame or Remote Frame) received.

Reception of Data Frame

The Message Handler FSM stores the message from the CAN Core shift register into the respective Message Object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is implemented to keep the data bytes connected with the identifier even if arbitration mask registers are used.

The NEWDAT bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset NEWDAT when it reads the Message Object. If at the time of the reception the NEWDAT bit was already set, MSGLST is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RXIE bit is set, the INTPND bit is set, causing the Interrupt Register to point to this Message Object.

The TXRQST bit of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

Reception of Remote Frame

When a Remote Frame is received, three different configurations of the matching Message Object have to be considered:

1. DIR = '1' (direction = transmit), RMTEN = '1', UMASK = '1' or '0'

   At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is set. The rest of the Message Object remains unchanged.
2. DIR = '1' (direction = transmit), RMTEN = '0', UMASK = '0'

   At the reception of a matching Remote Frame, the TXRQST bit of this Message Object remains unchanged, the Remote Frame is ignored.
3. DIR = '1' (direction = transmit), RMTEN = '0', UMASK = '1'

   At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is reset. The arbitration and control field (Identifier + IDE + RTR + DLC) from the shift register is stored into the Message Object in the Message RAM and the NEWDAT bit of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

Receive/Transmit Priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 (the highest implemented message object number) has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message Object.

Configuration of a Transmit Object

Figure 30-32 shows how a Transmit Object should be initialised.

Figure 30-32. Initialisation of a Transmit Object

MSGVAL	ARB	DATA	MASK	EOB	DIR	NEWDAT	MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

The Arbitration Registers (ID28-0 and XTD bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier ('Standard Frame') is used, it is programmed to ID28 - ID18, ID17 - ID0 can then be disregarded.

If the TXIE bit is set, the INTPND bit will be set after a successful transmission of the Message Object.

If the RMTEN bit is set, a matching received Remote Frame will cause the TXRQST bit to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Registers (DLC3-0, Data0-7) are given by the application, TXRQST and RMTEN may not be set before the data is valid.

The Mask Registers (MSK28-0, UMASK, MXTD, and MDIR bits) may be used (UMASK = '1') to allow groups of Remote Frames with similar identifiers to set the TXRQST bit. For details see section Transmission of Messages. Handle with care. The DIR bit should not be masked.

Updating a Transmit Object

The CPU may update the data bytes of a Transmit Object any time via the IFx Interface registers, neither MSGVAL nor TXRQST have to be reset before the update.

Note:

When the lowest priority message buffer is used for transmission, setting TXRQST = '0' may cause a delay of transmission, when TXRQST is set to '1' again. Please refer to the note at the description of TXRQST in 30.2.4.2 Transmission Request Registers (TREQR1n, TREQR2n).

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding IFx Data A Register or IFx Data B Register have to be valid before the content of that register is transferred to the Message Object. Either the CPU has to write all four bytes into the IFx Data Register or the Message Object is transferred to the IFx Data Register before the CPU writes the new data bytes. When only the (eight) data bytes are updated, first 0x0087 is written to the Command Mask Register and then the number of the Message Object is written to the Command Request Register, concurrently updating the data bytes and setting TXRQST.

To prevent the reset of TXRQST at the end of a transmission that may already be in progress while the data is updated, NEWDAT has to be set together with TXRQST. For details see Transmission of Messages.

When NEWDAT is set together with TXRQST, NEWDAT will be reset as soon as the new transmission has started.

Configuration of a Receive Object

Figure 30-33 shows how a Receive Object should be initialised.

Figure 30-33. Initialisation of a Receive Object

MSGVAL	ARB	DATA	MASK	EOB	DIR	NEWDAT	MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The Arbitration Registers (ID28-0 and XTD bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier ('Standard Frame') is used, it is programmed to ID28 - ID18, ID17 - ID0 can then be disregarded. When a Data Frame with an 11-bit Identifier is received, ID17 - ID0 will be set to '0'.

If the RXIE bit is set, the INTPND bit will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0) is given by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by non specified values.

The Mask Registers (MSK28-0, UMASK, MXTD, and MDIR bits) may be used (UMASK = '1') to allow groups of Data Frames with similar identifiers to be accepted. For details see Reception of Data Frame. The DIR bit should not be masked in typical applications.

Handling of Received Messages

The CPU may read a received message any time via the IFx Interface registers, the data consistency is guaranteed by the Message Handler state machine.

Typically the CPU will write first 0x007F to the Command Mask Register and then the number of the Message Object to the Command Request Register. That combination will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits NEWDAT and INTPND are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages has been received.

The actual value of NEWDAT shows whether a new message has been received since last time this Message Object was read. The actual value of MSGLST shows whether more than one message has been received since last time this Message Object was read. MSGLST will not be automatically reset.

By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TXRQST bit of a receive object will cause the transmission of a Remote Frame with the receive object's identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TXRQST bit is automatically reset.

Configuration of a FIFO Buffer

With the exception of the EOB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see Configuration of a Receive Object.

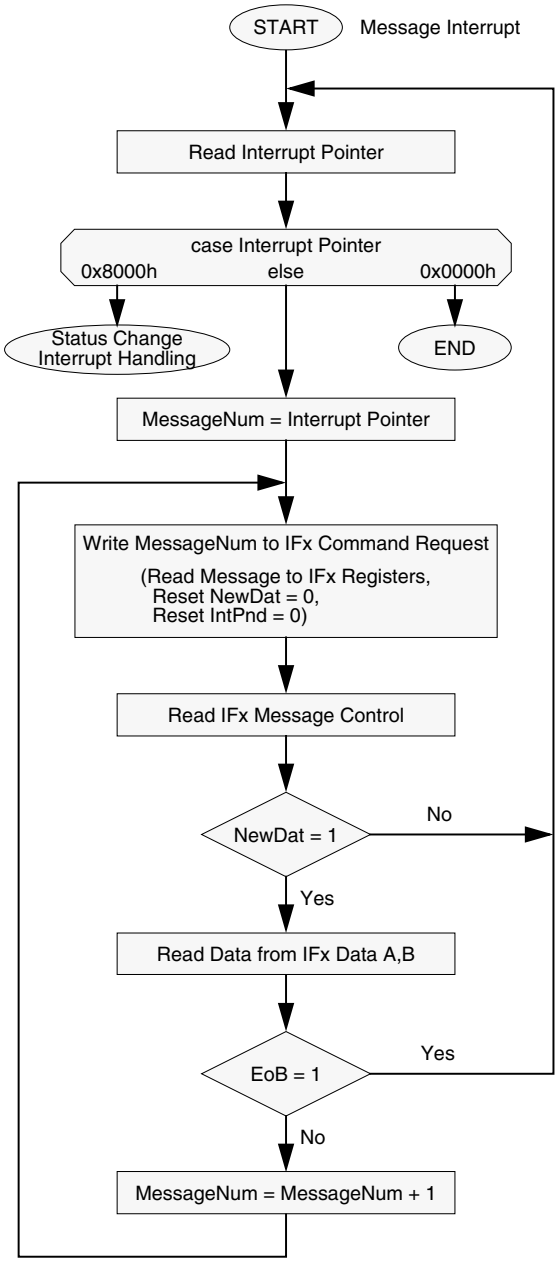
To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The EOB bit of all Message Objects of a FIFO Buffer except the last have to be programmed to zero. The EOB bit of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

Reading from a FIFO Buffer

When the CPU transfers the contents of Message Object to the IFx Message Buffer registers by writing its number to the IFx Command Request Register, the corresponding Command Mask Register should be programmed the way that bits NEWDAT and INTPND are reset to zero (TXRQST = '1' and CIP = '1'). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

To assure the correct function of a FIFO Buffer, the CPU should read out the Message Objects starting at the FIFO Object with the lowest message number. Figure 30-34 shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 30-34. CPU Handling of a FIFO Buffer



Handling of Interrupts

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, the Message Object's interrupt priority decreases with increasing message number.

A message interrupt is cleared by clearing the Message Object's INTPND bit. The Status Interrupt is cleared by reading the Status Register.

The interrupt identifier INTID in the Interrupt Register INTRn indicates the cause of the interrupt. When no interrupt is pending, the register will hold the value zero. If the value of the Interrupt Register is different from zero, then there is an interrupt

pending and, if IE is set, the interrupt line to the CPU is active. The interrupt line remains active until the Interrupt Register is back to value zero (the cause of the interrupt is reset) or until IE is reset.

The value 0x8000 indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits RXOK, TXOK and LEC, but a write access of the CPU to the Status Register can never generate or reset an interrupt.

All other values indicate that the source of the interrupt is one of the Message Objects, INTID points to the pending message interrupt with the highest interrupt priority.

The CPU controls whether a change of the Status Register may cause an interrupt (bits EIE and SIE in the CAN Control Register) and whether the interrupt line becomes active when the Interrupt Register is different from zero (bit IE in the CAN Control Register). The Interrupt Register will be updated even when IE is reset.

The CPU has two possibilities to follow the source of a message interrupt. First it can follow the INTID in the Interrupt Register and second it can poll the Interrupt Pending Register (see 30.2.4 Message Handler Registers).

An interrupt service routine reading the message that is the source of the interrupt may read the message and reset the Message Object's INTPND at the same time (bit CIP in the Command Mask Register IFxCMSKn). When INTPND is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1kBit/s up to 1000kBit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods (fosc) may be different.

The frequencies of these oscillators are not absolutely stable, small variations are caused by changes in temperature or voltage and by deteriorating components.

As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronising to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 30-35). The Synchronisation Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 1). The length of the time quantum (tq), which is the basic time unit of the bit time, is defined by the CAN controller's system clock fsys and the Baud Rate Prescaler (BRP) : tq = BRP /fsys. The CAN's system clock fsys is the frequency of its CLK_PERI1_PD2 input.

The Synchronisation Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment Prop_Seg is intended to compensate for the physical delay times within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronisation Jump Width (SJW) defines how far a resynchronisation may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

Figure 30-35. Bit Timing

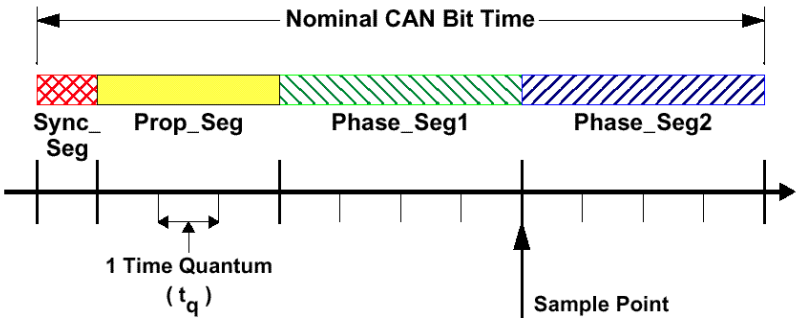


Table 30-35. Parameters of the CAN Bit Time

Parameter	Range	Remark
BRP	[1..32]	defines the legth of the time quantum tq
Sync_Seg	1 tq	fixed length, synchronisation of us into system clock
Prop_Seg	[1..8] tq	compensates for the physical delay times
Phase_Seg1	[1..8] tq	may be lengthened temporarily by synchronisation
Phase_Seg2	[1..8] tq	may be shortened temporarily by synchronisation
SJW	[1..4] tq	may not be longer than either Phase Buffer Segment

Note:

This table describes the minimum programmable ranges required by the CAN protocol. As described earlier in this chapter for BTRn register fields, programming is as follow:

Prop_Seg + Phase_Seg1 = TSEG1 value +1

Phase_Seg2 = TSEG2 value + 1

31. LIN-USART

This chapter explains the function and operation of the LIN-USART.

31.1 Overview of the LIN-USART

This section describes the features and the block diagram of the LIN-USART.

Features of LIN-USART

The LIN-USART with LIN (Local Interconnect Network) - function is a general-purpose serial data communication interface for performing synchronous or asynchronous communication with external devices. The LIN-USART provides bidirectional communication function (normal mode), master- slave communication function (multiprocessor mode in master/slave systems) and special features for operating as a LIN master in a LIN-bus systems.

See Table 31-1 for more details of the functions.

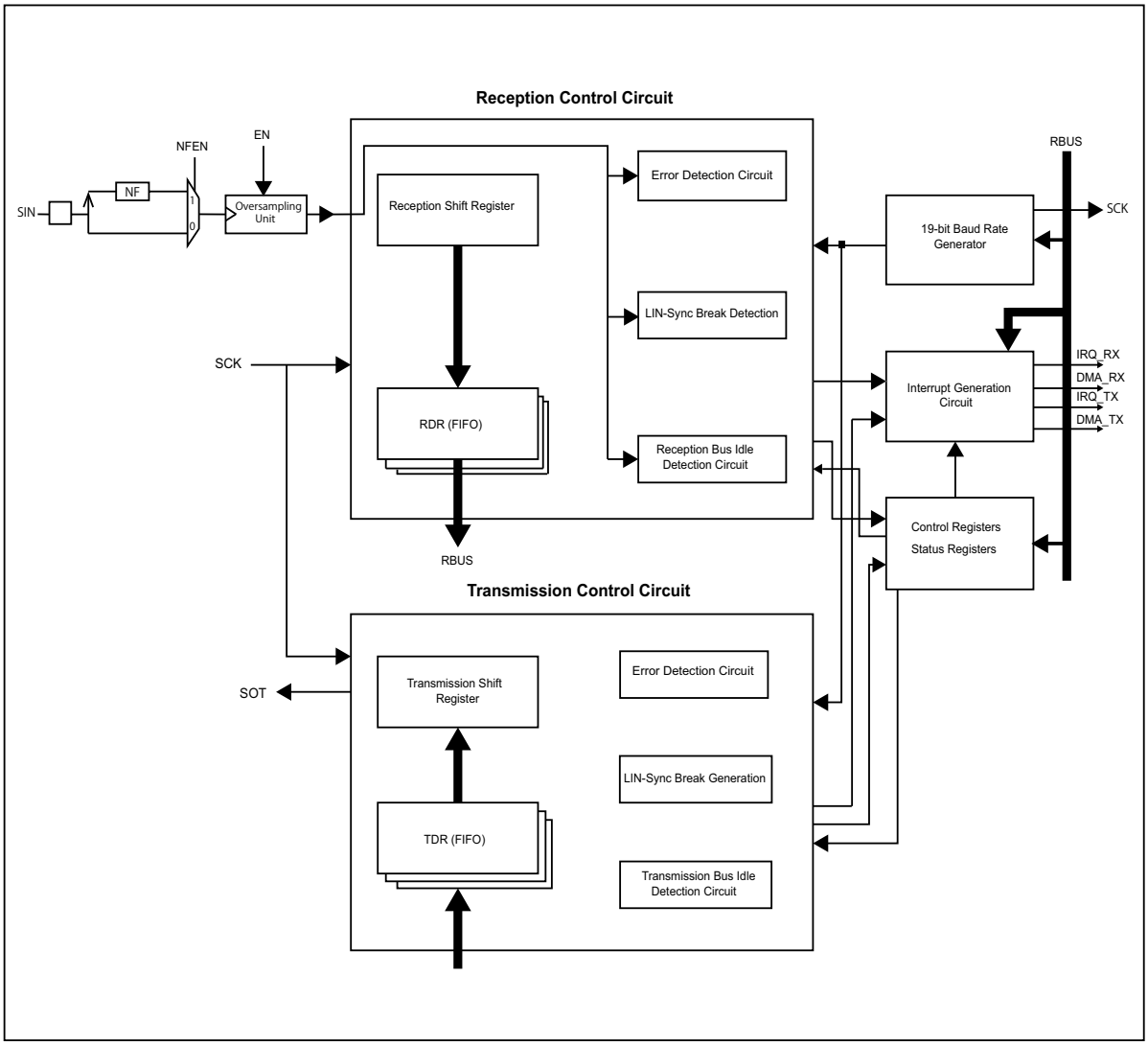
Table 31-1. LIN-USART functions

Item	Function
Data buffer	Full-duplex
Serial input	Five times oversampling in asynchronous mode
Transfer mode	■ Clock synchronous (start-stop synchronization and start- stop bit option) ■ Clock asynchronous (using start and stop bits)
Baud rate	■ A dedicated baud rate generator is provided, which consists of a 19-bit reload counter ■ An external clock can be input and also be adjusted by the reload counter
Data length	■ 7 bits (not in synchronous or LIN mode) ■ 8 bits
Signal mode	■ Normal data mode ■ Inverted data mode
Start bit timing	Clock synchronization to the falling edge of the start bit in asynchronous mode (normal mode) respective of the rising edge in inverted mode
Reception error detection	■ Framing error ■ Overrun error ■ Parity error in normal mode
16 byte FIFO	■ Independent 16 byte FIFO for transmission and reception ■ FIFO can be activated with transmit/receive trigger level for interrupt

Table 31-1. LIN-USART functions

Item	Function
Interrupt request	■ Reception interrupt (reception data register full, reception FIFO trigger level reached) ■ Transmission interrupt (transmission data register empty, transmission FIFO trigger level reached, last bit being shifted out and bus idle) ■ Error interrupt (reception error detected and transmission error detected) ■ Transmission and reception DMA request support
Master-slave communication function (multiprocessor mode)	One-to-n communication (one master to n slaves) (This function is supported both for master and slave system)
Synchronous mode	Function as master or slave LIN-USART
Transceiving pins	Direct access possible
LIN bus options	■ Operation as master device ■ Generation of LIN sync break ■ Detection of physical bus error ■ Internal loop back feature ■ Transmission/reception FIFO of configurable depth 1 to 16 bytes
Synchronous serial clock	The synchronous serial clock can be output continuously on the SCK pin for synchronous communication with start and stop bits
Clock delay option	Special synchronous clock mode for delaying clock by 1/2 clock phase (useful for SPI)
Debug mode	■ Debug mode support is provided by LIN-USART ■ Debugger can read the received data from USARTn_RDR register.

Figure 31-1. Block diagram of LIN-USART



31.1.1 Block diagram of LIN-USART

The LIN-USART consists of the following blocks:

- Reception Control Circuit
  - Reception Shift Register
  - Reception Data Register (USARTn_RDR) with FIFO
  - Error Detection Circuit
  - Reception Bus Idle Detection Circuit
  - LIN-SYNC Break Detection Circuit
- Transmission Control Circuit
  - Transmission Shift Register
  - Transmission Data Register (USARTn_TDR) with FIFO
  - Error Detection Circuit
  - Transmission Bus Idle Detection Circuit
  - LIN-SYNC Break Generation Circuit
- Oversampling Unit
- Interrupt Generation Circuit
- Baud Rate Generation circuit
- Control and Status Registers

Reception Control Circuit

The reception control circuit consists of a received bit counter, start bit detection circuit and received parity counter. The received bit counter counts reception data bits. When reception of one data frame for the specified data length is complete, the received bit counter sets the reception data register full flag. The start bit detection circuit detects start bits from the serial input signal and sends a signal to the reload counter to synchronize it to the falling edge of these start bits. The reception parity counter calculates the parity of the reception data.

- Reception Shift Register

  The reception shift register fetches reception data input from the SIN pin, shifting the data bit by bit. When reception is complete, the reception shift register transfers receive data to the USARTn_RDR register.
- Reception Data Register (USARTn_RDR) with FIFO

  This register retains reception data. Serial data input is converted and stored in this register. The reception FIFO is 16 bytes and is used for storing the data during reception.
- LIN-Sync Break Detection Circuit

  The LIN Break Detection Circuit detects a LIN break of a determined length.

Transmission Control Circuit

The Transmission Control Circuit consists of a transmission bit counter, transmission start circuit, and transmission parity counter. The transmission bit counter counts the transmission data bits. When the transmission of one data frame of the specified data length is complete, the transmission bit counter sets the Transmission Data Register empty flag. The transmission start circuit starts transmission when data is written to USARTn_TDR. The transmission parity counter generates a parity bit for data to be transmitted if parity is enabled.

- Transmission Shift Register

  The transmission shift register loads data written to the USARTn_TDR and outputs the data to the SOT pin, shifting the data bit by bit.
- Transmission Data Register (USARTn_TDR) with FIFO

  This register sets transmission data. Data written to this register is converted to serial data and is output to SOT. The transmission FIFO is 16 bytes. It is used for storing the data during transmission.
- LIN-Sync Break Generation Circuit

  The LIN Break Generation Circuit generates a LIN break of a determined length.

Common to both Reception/Transmission Control Circuits

- Error Detection Circuit

  The Error Detection Circuit checks if there was any error during the last reception/transmission. If an error has occurred it sets the corresponding error flags.
- Bus Idle Detection Circuit

  The Reception Bus Idle Detection Circuit recognizes if neither a reception/transmission is ongoing. In this case, the circuit sets the corresponding USARTn_ECCR:RBI/USARTn_ECCR:TBI bits.

Oversampling Unit

The Oversampling Unit oversamples the incoming data at the SIN pin by a factor of five. It is switched off in synchronous operation mode.

Interrupt Generation Circuit

The Interrupt Generation Circuit administers all cases of generating a reception or transmission or error interrupt. If a corresponding enable flag is set and an interrupt case occurs the interrupt will be generated immediately.

Baud Rate Generation Circuit

The Baud Rate Generation Circuit generates a specified baud rate.

Status and Control Register

The Status and Control Registers provide the status and control of a number of user-configurable features.

31.2 LIN-USART registers

This section describes the registers of the LIN-USART in detail.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of LIN-USART

The following registers are available for each instance of LIN-USART:

- Serial Mode Register (USARTn_SMR)
- Serial Control Register (USARTn_SCR)
- Serial Mode Set Register (USARTn_SMSR)
- Serial Control Set Register (USARTn_SCSR)
- Serial Control Clear Register (USARTn_SCCR)
- Transmission Data Register (USARTn_TDR)
- Serial Status Register (USARTn_SSR)
- Reception Data Register (USARTn_RDR)
- Serial Status Set Register (USARTn_SSSR)
- Serial Status Clear Register (USARTn_SSCR)
- Extended Communication Control Register (USARTn_ECCR)
- Extended Status/Control Register (USARTn_ESCR)
- Extended Communication Control Set Register (USARTn_ECCSR)
- Extended Status/Control Set Register (USARTn_ESCSR)
- Extended Communication Control Clear Register (USARTn_ECCCR)
- Extended Status/Control Clear Register (USARTn_ESCCR)
- Extended Serial Interrupt Register (USARTn_ESIR)
- Extended Interrupt Enable Register (USARTn_EIER)
- Extended Serial Interrupt Set Register (USARTn_ESISR)
- Extended Interrupt Enable Set Register (USARTn_EIESR)
- Extended Serial Interrupt Clear Register (USARTn_ESICR)
- Extended Interrupt Enable Clear Register (USARTn_EIECR)
- Extended Feature Enable Register - L (USARTn_EFERL)
- Extended Feature Enable Register - H (USARTn_EFERH)
- Reception FIFO Control Register (USARTn_RFCR)
- Transmission FIFO Control Register (USARTn_TFCR)
- Reception FIFO Control Set Register (USARTn_RFCSR)
- Transmission FIFO Control Set Register (USARTn_TFCSR)
- Reception FIFO Control Clear Register (USARTn_RFCCR)
- Transmission FIFO Control Clear Register (USARTn_TFCCR)
- Reception FIFO Status Register (USARTn_RFSR)
- Transmission FIFO Status Register (USARTn_TFSR)
- Extended Status Register (USARTn_ESR)
- Extended Status Clear Register (USARTn_ESCLR)
- Baud Rate Generation Reload Register (USARTn_BGRLL)
- Baud Rate Generation Register (USARTn_BGRL)
- Serial Transmit DMA Configuration Register (USARTn_STXDR)
- Serial Receive DMA Configuration Register (USARTn_SRXDR)
- Serial Transmit DMA Configuration Set Register (USARTn_STXDSR)

- Serial Receive DMA Configuration Set Register (USARTn_SRXDSR)
- Serial Transmit DMA Configuration Clear Register (USARTn_STXDCR)
- Serial Receive DMA Configuration Clear Register (USARTn_SRXDCR)
- Debug Register (USARTn_DEBUG)
- Unused Registers

Memory layout of LIN-USART registers

Table 31-2. Memory layout of LIN-USART registers

Offset	+1	+0
0x00000000	USARTn_SCR 00000000	USARTn_SMR 00000000
0x00000002	USARTn_SCSR 00000000	USARTn_SMSR 00000000
0x00000004	USARTn_SCCR 00000000	reserved XXXXXXXX
0x00000006	USARTn_SSR 00001000	USARTn_TDR 00000000
0x00000008	USARTn_SSSR 00000000	USARTn_RDR 00000000
0x0000000A	USARTn_SSCR 00000000	reserved XXXXXXXX
0x0000000C	USARTn_ESCR 00000100	USARTn_ECCR 000000XX
0x0000000E	USARTn_ESCSR 00000000	USARTn_ECCSR 00000000
0x00000010	USARTn_ESCCR 00000000	USARTn_ECCCR 00000000
0x00000012	USARTn_EIER 00000000	USARTn_ESIR 000010X0
0x00000014	USARTn_EIESR 00000000	USARTn_ESISR 00000000
0x00000016	USARTn_EIECR 00000000	USARTn_ESICR 00000000
0x00000018	USARTn_EFERH 00000000	USARTn_EFERL 00000000
0x0000001A	USARTn_TFCR 00000000	USARTn_RFCR 00000000
0x0000001C	USARTn_TFCSR 00000000	USARTn_RFCSR 00000000
0x0000001E	USARTn_TFCCR 00000000	USARTn_RFCCR 00000000

Table 31-2. Memory layout of LIN-USART registers

Offset	+1	+0
0x00000020	USARTn_TFSR 00000000	USARTn_RFSR 00000000
0x00000022	USARTn_ESR 00000000	USARTn_CSCR 00000000
0x00000024	reserved XXXXXXXX	USARTn_CSCSR 00000000
0x00000026	USARTn_ESCLR 00000000	USARTn_CSCCR 00000000
0x00000028	USARTn_BGRLM 00000000	USARTn_BGRLL 00000000
0x0000002A	reserved XXXXXXXX	USARTn_BGRLH 00000000
0x0000002C	USARTn_BGRM 00000000	USARTn_BGRL 00000000
0x0000002E	reserved XXXXXXXX	USARTn_BGRH 00000000
0x00000030	USARTn_SRXDR 00000000	USARTn_STXDR 00000000
0x00000032	USARTn_SRXDSR 00000000	USARTn_STXDSR 00000000
0x00000034	USARTn_SRXDCR 00000000	USARTn_STXDCR 00000000
0x00000036	read0 00000000	read0 00000000
0x00000038	reserved XXXXXXXX	read0 00000000
0x0000003A	reserved XXXXXXXX	USARTn_FIDR 00000000
0x0000003C	reserved XXXXXXXX	USARTn_DEBUG 00000000

31.2.1 Serial Mode Register (USARTn_SMR)

This register selects the operation mode and baud rate clock, and allows the resetting of the complete LIN-USART and baud rate generator.

Serial Mode Register (USARTn_SMR)

Figure 31-2. Serial Mode Register (USARTn_SMR)

USARTn_SMR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MD[1]	MD[0]	OTO	EXT	REST	UPCL	read0	NFEN
RpWp	RpWp	RpWp	RpWp	Rp0Wp1	Rp0Wp1	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 31-3. Serial Mode Register (USARTn_SMR) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	MD	Operation Mode Selection These two bits set the LIN-USART operation mode. '00': Mode 0: Asynchronous normal '01': Mode 1: Asynchronous multiprocessor '10': Mode 2: Synchronous '11': Mode 3: Asynchronous LIN Changing the mode bits causes any ongoing reception or transmission to be aborted. Changing the mode also sets USARTn_SSR:TDRE and USARTn_ESIR:TDRE and clears USARTn_SSR:RDRF. It is strongly recommended to reset the USART (write '1' to USARTn_SMR:UPCL) after changing operation mode.
[5]	OTO	One-to-One External Clock Selection This bit sets an external clock (SCKn) to match the LIN-USART's serial clock. This function is used for operating Mode 2 (synchronous) slave mode operation. '0': Use external clock with baud rate generator (reload counter) '1': Use external clock as it is **Note:** USARTn_SMR:OTO is masked with USARTn_SMR:EXT. This means that If USARTn_SMR:EXT = '0', the value read from USARTn_SMR:OTO will also be '0' regardless of what value was written to it.

Table 31-3. Serial Mode Register (USARTn_SMR) bits

Bit Position	Bit Field Name	Bit Description
[4]	EXT	External Clock Selection This bit selects the internal or external clock source for the reload counter. '0': Use internal baud rate generator (reload counter) '1': Use external serial clock source
[3]	REST	Restart of Transmission Reload Counter '0': No effect '1': Reload counter is restarted Reading from this bit always returns '0'.
[2]	UPCL	LIN-USART Programmable Clear (software reset) '0': No effect '1': Resets the LIN-USART immediately. The register settings are preserved. Possible reception or transmission will be cut off This bit also resets the TX and RX FIFO block. All flags are cleared and the Reception Data Register (USARTn_RDR) contains 0x00. Reading this bit always returns '0'. A LIN-USART reset should be performed after disabling reception and transmission, and the LIN_USART interrupt enable bits.
[1]	read0	-
[0]	NFEN	Noise Filter Enable for SIN pin '0': Analog noise filter input is not used '1': Analog noise filter input is used

31.2.2 Serial Control Register (USARTn_SCR)

This register specifies parity bits, selects the stop bit and data lengths, selects a frame data format in Mode 1, clears the reception error flags, and specifies whether transmission and reception should be enabled.

Serial Control Register (USARTn_SCR)

Figure 31-3. Serial Control Register (USARTn_SCR)

USARTn_SCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PEN	P	SBL	CL	AD	CRE	RXE	TXE
RpWp	RpWp	RpWp	RpWp	RpWp	Rp0Wp1	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-4. Serial Control Register (USARTn_SCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	PEN	Parity Enable This bit selects whether to add a parity bit during transmission or detect it during reception. Parity is provided in Mode 0 and in Mode 2 if USARTn_ECCR:SSM = '1'. This bit is fixed to '0' (no parity) in Mode 1 (multiprocessor) and Mode 3 (LIN). '0': Parity disabled '1': Parity enabled
[6]	P	Parity Selection When parity is provided and enabled this bit selects even or odd parity. '0': Even parity enabled '1': Odd parity enabled
[5]	SBL	Stop Bit Length Selection This bit selects the length of the stop bit of an asynchronous data frame or a synchronous frame if USARTn_ECCR:SSM = '1'. This bit is fixed to '0' (1 stop bit) in Mode 3 (LIN). '0': One (1) stop bit '1': Two (2) stop bits

Table 31-4. Serial Control Register (USARTn_SCR) bits

Bit Position	Bit Field Name	Bit Description
[4]	CL	Character (Data frame) Length This bit specifies the length of the data transmitted or received data. It is fixed to '1' (8 bits) in Modes 2 and 3. '0': Seven (7) bits '1': Eight (8) bits
[3]	AD	Address or Data Selection This bit sets the AD status of transmitted frame in multiprocessor Mode 1. '0': Indicates a normal data frame '1': Indicates an address frame Reading this bit returns the AD status of the transmitted frame in multiprocessor Mode 1.
[2]	CRE	Clear Reception Error Flags This bit clears the USARTn_SSR:FRE, USARTn_SSR:ORE and USARTn_SSR:PE flags. It can also reset any ongoing reception depending on USARTn_EFERL:RSTRFM (default is reset). '0': No effect '1': Clears the error flags Reading this bit always returns '0'.
[1]	RXE	Reception Enable This bit enables or disables LIN-USART reception. '0': LIN-USART disables the reception of data frames '1': LIN-USART enables the reception of data frames **Notes:** 1. If RXE is set to '0' during reception, reception is stopped immediately. In this case data is not guaranteed. 2. The LIN sync break detection in Mode 3 is enabled independend of the USARTn_SCR:RXE setting. 3. RXE is always enabled if the internal loop back is enabled (i.e. USARTn_EFERL:INTLBEN='1'). Read value of USARTn_SCR:RXE is then '1'.
[0]	TXE	Transmission Enable This bit enables or disables LIN-USART transmission. '0': LIN-USART disables the transmission of data frames '1': LIN-USART enables the transmission of data frames **Note:** If TXE is set to '0' during transmission, transmission is stopped immediately. In this case data is not guaranteed.

31.2.3 Serial Mode Set Register (USARTn_SMSR)

This register sets the bits of Serial Mode Register.

Serial Mode Set Register (USARTn_SMSR)

Figure 31-4. Serial Mode Set Register (USARTn_SMSR)

USARTn_SMSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	RESTS	UPCLS	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-5. Serial Mode Set Register (USARTn_SMSR) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	RESTS	REST Set '0': No effect '1': Sets the USARTn_SMR:REST bit Reading this bit always returns '0'.
[2]	UPCLS	UPCL Set '0': No effect '1': Sets the USARTn_SMR:UPCL bit Reading this bit always returns '0'.
[1:0]	read0	-

31.2.4 Serial Control Set Register (USARTn_SCSR)

This register sets the bits of Serial Control Register.

Serial Control Set Register (USARTn_SCSR)

Figure 31-5. Serial Control Set Register (USARTn_SCSR)

USARTn_SCSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	ADS	CRES	RXES	TXES
Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-6. Serial Control Set Register (USARTn_SCSR) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	ADS	AD Set '0': No effect '1': Sets the AD bit of the transmitted frame, which indicates that it is an address frame Reading this bit always returns '0'.
[2]	CRES	CRE Set '0': No effect '1': Sets the USARTn_SCR:CRE bit Reading this bit always returns '0'.
[1]	RXES	RXE Set '0': No effect '1': Sets the USARTn_SCR:RXE bit Reading this bit always returns '0'.
[0]	TXES	TXE Set '0': No effect '1': Sets the USARTn_SCR:TXE bit Reading this bit always returns '0'.

31.2.5 Serial Control Clear Register (USARTn_SCCR)

This register clears the bits of Serial Control Register.

Serial Control Clear Register (USARTn_SCCR)

Figure 31-6. Serial Control Clear Register (USARTn_SCCR)

USARTn_SCCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	ADC	read0	RXEC	TXEC
Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-7. Serial Control Clear Register (USARTn_SCCR) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	ADC	AD Clear '0': No effect '1': Clears the AD bit of the transmitted frame, which indicates that it is a data frame Reading this bit always returns '0'.
[2]	read0	-
[1]	RXEC	RXE Clear '0': No effect '1': Clears the USARTn_SCR:RXE bit Reading this bit always returns '0'.
[0]	TXEC	TXE Clear '0': No effect '1': Clears the USARTn_SCR:TXE bit Reading this bit always returns '0'.

31.2.6 Transmission Data Register (USARTn_TDR)

The Transmission Data Register (USARTn_TDR) holds the transmission data.

Transmission Data Register (USARTn_TDR)

Figure 31-7. Transmission Data Register (USARTn_TDR)

USARTn_TDR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0

Table 31-8. Transmission Data Register (USARTn_TDR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	D	Transmission Data Reading this register always returns '0'.

Transmission:

When transmission data is written to this register, the transmission data empty flag bit (USARTn_SSR:TDRE) is cleared to '0'.

When transfer to the transmission shift register has completed, USARTn_SSR:TDRE is set to '1'. When the USARTn_SSR:TDRE bit is '1', the next part of transmission data can be written. If transmission interrupt requests have been enabled, a transmission interrupt is generated.

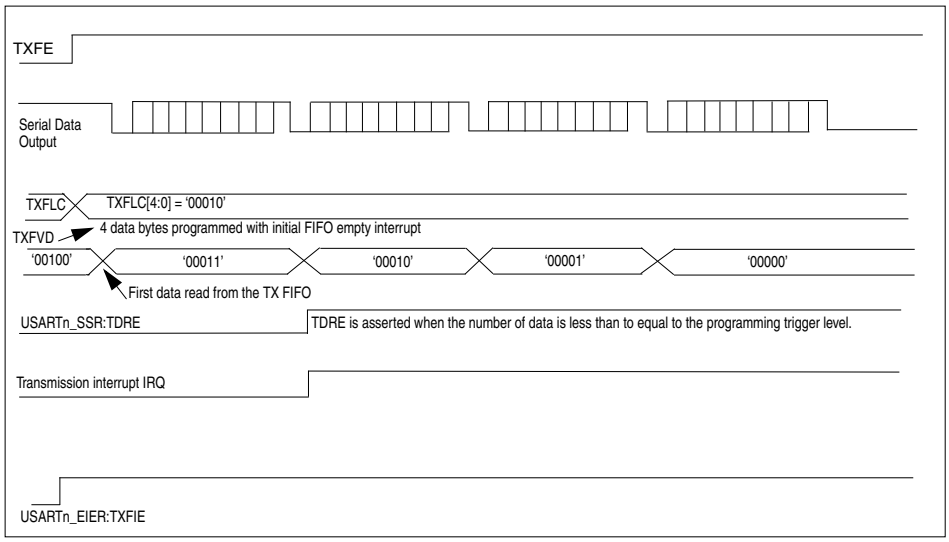
Write the next part of transmission data when a transmission interrupt is generated or the USARTn_SSR:TDRE bit is '1'.

If the TX FIFO is enabled (USARTn_TFCR:TXFE = '1'), USARTn_SSR:TDRE is set when the number of data in the TX FIFO is less or equal to programmed trigger level in the USARTn_TFCR:TXFLC[4:0].

In this case transmission interrupt is asserted only if USARTn_EIER:TXFIE is set.

USARTn_TDR and TX FIFO-content is reset to '11111111' at reset.

Figure 31-8. Transmission of LIN-USART with FIFO



31.2.7 Serial Status Register (USARTn_SSR)

This register shows the transmission and reception status. It also enables and disables the transmission and reception interrupts.

Serial Status Register (USARTn_SSR)

Figure 31-9. Serial Status Register (USARTn_SSR)

USARTn_SSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE
Rp	Rp	Rp	Rp	Rp	RpWp	RpWp	RpWp
0	0	0	0	1	0	0	0

Table 31-9. Serial Status Register (USARTn_SSR) bits

Bit Position	Bit Field Name	Bit Description
[7]	PE	Parity Error Flag '0': No parity error occurred '1': A parity error occurred during reception This bit is cleared when '1' is written to USARTn_SCR:CRE. When this bit and the USARTn_ESIR:PEIE bit are '1', an error interrupt request is generated. If RX FIFO is disabled (USARTn_RFCR:RXFE = '0') and this flag is set, data in the Reception Data Register (USARTn_RDR) is invalid. If RX FIFO is enabled (USARTn_RFCR:RXFE = '1'), the error interrupt request must be dealt with before the next data is received otherwise invalid data cannot be identified and all the data must be thrown away.

Table 31-9. Serial Status Register (USARTn_SSR) bits

Bit Position	Bit Field Name	Bit Description
[6]	ORE	Overrun Error Flag '0': No overrun error occurred '1': An overrun error occurred during reception This bit is cleared when '1' is written to the USARTn_SCR:CRE. When this bit and USARTn_ESIR:OREIE are '1', an error interrupt request is generated. When this flag is set, data in the Reception Data Register (USARTn_RDR) is invalid. If RX FIFO is enabled (USARTn_RFCR:RXFE = '1'), this bit is set when the received FIFO is full (i.e. all 16 bytes) and another data byte is received.
[5]	FRE	Framing Error Flag '0': No framing error occurred '1': A framing error occurred during reception This bit is cleared when '1' is written to USARTn_SCR:CRE or USARTn_SMR:UPCL. When this bit and USARTn_ESIR:FREIE are '1', an error interrupt request is generated. If RX FIFO is disabled (USARTn_RFCR:RXFE = '0') and this flag is set, data in the Reception Data Register (USARTn_RDR) is invalid. If RX FIFO is enabled (USARTn_RFCR:RXFE = '1'), the error interrupt request must be dealt with before the next data is received otherwise invalid data cannot be identified and all the data must be thrown away.

Table 31-9. Serial Status Register (USARTn_SSR) bits

Bit Position	Bit Field Name	Bit Description
[4]	RDRF	Receive Data Register Full Flag This flag indicates the status of the Reception Data Register (USARTn_RDR). '0': Reception Data Register is empty '1': Reception Data Register is full This bit is set to '1' when reception data is loaded into USARTn_RDR. When RX FIFO is enabled (USARTn_RFCR:RXFE = '1') this bit is set to '1' when the number of data in the RX FIFO reaches the trigger level programmed in USARTn_RFCR:RXFLC[4:0]. It is cleared when: ■ The Reception Data Register (USARTn_RDR) is read ■ '1' is writen to USARTn_SMR:UPCL or USART operation mode is changed (USARTn)SMR:MD[1:0] ■ When RX FIFO is disabled (i.e. USARTn_RFCR:RXFE = '0'), a reception interrupt request is output if both this bit and USARTn_SSR:RIE are '1' ■ When RX FIFO is enabled (i.e. USARTn_RFCR:RXFE = '1'), a reception interrupt request is output if both this bit and USARTn_EIER:RXFIE are '1' **Notes:** 1. The RDRF flag is set under the conditions mentioned above only if no reception error occured and all reception errors are cleared. 2. USARTn_ESIR:RDRF has the same behaviour as this bit but it is not cleared when the Reception Data Register (USARTn_RDR) is read.

Table 31-9. Serial Status Register (USARTn_SSR) bits

Bit Position	Bit Field Name	Bit Description
[3]	TDRE	Transmission Data Register Empty This flag indicates the status of the Transmission Data Register (USARTn_TDR). '0': Transmission Data Register is full '1': Transmission Data Register is empty When TX FIFO is enabled (USARTn_TFCR:TXFE = '1') the TDRE flag is set to '1' when the number of data in the TX FIFO reaches the programmed trigger level USARTn_TFCR:TXFLC[4:0]. This bit is cleared to '0' when transmission data is written to USARTn_TDR. It is set to '1' when: ■ Data is loaded into the Transmission Shift Register and transmission starts ■ A '1' is written to USARTn_SMR:UPCL ■ USART operation mode is changed (USARTn_SMR:MD[1:0]) When TX FIFO is disabled (i.e. USARTn_TFCR:TXFE = '0'), a transmission interrupt request is output if both this bit and USARTn_SSR:TIE are '1'. When TX FIFO is enabled (i.e. USARTn_TFCR:TXFE = '1'), a transmission interrupt request is output if both this bit and USARTn_EIER:TXFIE are '1'. If the USARTn_ECCR:LBR is set to '1' while this bit is '1', it changes to '0'; after the completion of LIN sync break generator, this bit changes back to '1'. **Notes:** 1. This bit is set to '1' (USARTn_TDR empty) as its initial value. 2. USARTn_ESIR:TDRE has the same behaviour as this bit. But it is not cleared when transmission data is written to USARTn_TDR.
[2]	BDS	Transfer Direction Selection This bit selects whether to transfer serial data from the least significant bit (LSB) or from the most significant bit (MSB). '0': LSB first '1': MSB first **Notes:** 1. The high- and low-order sides of serial data are interchanged with each other during reading from or writing to the Serial Data Register. 2. If this bit is set to another value after the data is written to the USARTn_RDR register the data becomes invalid. 3. This bit is fixed to '0' in mode 3 (LIN mode).

Table 31-9. Serial Status Register (USARTn_SSR) bits

Bit Position	Bit Field Name	Bit Description
[1]	RIE	Receive Interrupt Request Enable This bit enables or disables the reception interrupt if RX-FIFO is disabled. If USARTn_SSR:RDRF bit is set and this bit is '1', then a reception interrupt is signaled to the Interrupt Controller (if RX-FIFO is disabled). '0': Disables reception interrupt '1': Enables reception interrupt if RX-FIFO is not used **Note:** If RX-FIFO is enabled (USARTn_RFCR:RXFE = '1'), USARTn_EIER:RXFIE is used to enable or disable the reception interrupt.
[0]	TIE	Transmission Interrupt Request Enable This bit enables or disables the transmission interrupt if TX-FIFO is disabled. If this bit and USARTn_SSR:TDRE are '1', then a transmission interrupt request is output (if TX-FIFO is disabled, i.e. USARTn_TFCR:TXFE = '0'). This interrupt can be used to replace the Last Bit Shifted Out Interrupt Enable bit (USARTn_EIER:LBSOIE). Only one of these interrupts should be enabled at the same time. '0': Disables transmission interrupt '1': Enables transmission interrupt **Note:** If TX-FIFO is enabled (i.e. USARTn_TFCR:TXFE = '1'), USARTn_EIER:TXFIE is used to enable or disable the transmission interrupt.

31.2.8 Reception Data Register (USARTn_RDR)

The Reception Data Register (USARTn_RDR) holds the received data.

Reception Data Register (USARTn_RDR)

Figure 31-10. Reception Data Register (USARTn_RDR)

USARTn_RDR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 31-10. Reception Data Register (USARTn_RDR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	D	Received Serial Data Stores the data received over the serial line.

Reception:

USARTn_RDR is the register that contains reception data. The serial data signal transmitted to the SINn pin is converted in the shift register and stored there. When the data length is 7 bits, the uppermost bit (D[7]) contains 0. When reception is complete the data is stored in this register and the reception data full flag bit (USARTn_SSR:RDRF) is set to '1'. If a receive interrupt request is enabled at this point, a receive interrupt occurs.

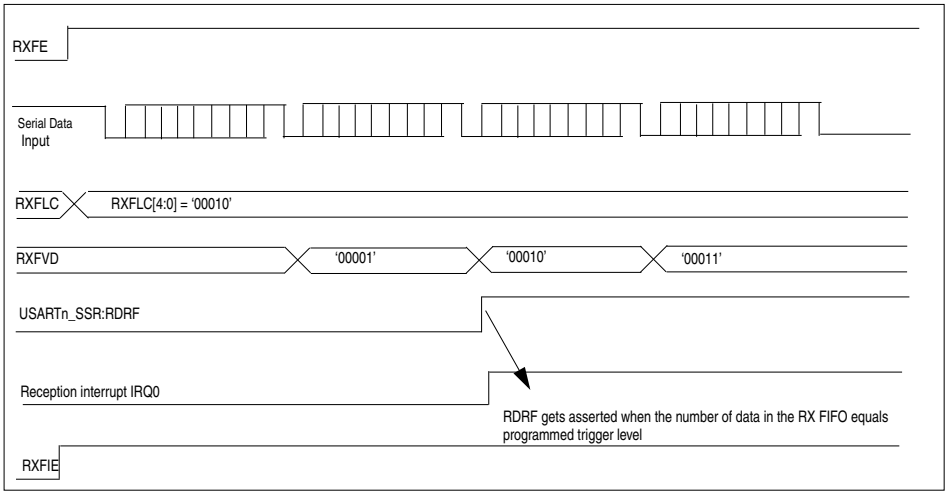
Read USARTn_RDR when USARTn_SSR:RDRF bit is '1'. USARTn_SSR:RDRF bit is cleared automatically when USARTn_RDR is read. Also the receive interrupt is cleared if it is enabled and no error has occurred.

Data in USARTn_RDR is invalid when a reception error occurs (USARTn_SSR:PE, USARTn_SSR:ORE, or USARTn_SSR:FRE is '1').

If the RX FIFO is enabled (USARTn_RFCR:RXFE = '1'), USARTn_RDR contains the next value of the RX FIFO to be read. USARTn_SSR:RDRF is set when the number of data in the FIFO is greater or equal to the programmed trigger level in the USARTn_RFCR:RXFLC[4:0]. In this case reception interrupt is asserted only if USARTn_EIER:RXFIE is also set.

USARTn_RDR and RX FIFO-content is reset to '00000000' at reset and USART software reset (USARTn_SMR:UPCL).

Figure 31-11. Reception of LIN-USART with FIFO



31.2.9 Serial Status Set Register (USARTn_SSSR)

This register sets the bits of Serial Status Register.

Figure 31-12. Serial Status Set Register (USARTn_SSSR)

USARTn_SSSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	RIES	TIES
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-11. Serial Status Set Register (USARTn_SSSR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	RIES	RIE Set '0': No effect '1': Sets the USARTn_SSR:RIE bit Reading this bit always returns '0'.
[0]	TIES	TIE Set '0': No effect '1': Sets the USARTn_SSR:TIE bit Reading this bit always returns '0'.

31.2.10 Serial Status Clear Register (USARTn_SSCR)

This register clears the bits of Serial Status Register.

Figure 31-13. Serial Status Clear Register (USARTn_SSCR)

USARTn_SSCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	RIEC	TIEC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-12. Serial Status Clear Register (USARTn_SSCR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	RIEC	RIE Clear '0': No effect '1': Clears the USARTn_SSR:RIE bit Reading this bit always returns '0'.
[0]	TIEC	TIE Clear '0': No effect '1': Clears the USARTn_SSR:TIE bit Reading this bit always returns '0'.

31.2.11 Extended Communication Control Register (USARTn_ECCR)

The Extended Communication Control Register provides data inversion, LIN break generation, Master/Slave select mode, bus idle recognition interrupt settings, and synchronous clock settings.

Extended Communication Control Register (USARTn_ECCR)

Figure 31-14. Extended Communication Control Register (USARTn_ECCR)

USARTn_ECCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
INV	LBR	MS	SCDE	SSM	BIE	RBI	TBI
RpWp	Rp0Wp1	RpWp	RpWp	RpWp	RpWp	Rp	Rp
0	0	0	0	0	0	X	X

Table 31-13. Extended Communication Control Register (USARTn_ECCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	INV	Invert Serial Data This bit inverts the serial data (including the Start and Stop bits) at the SIN and SOT pin. SCK is not affected (see USARTn_ESCR:SCES). '0': Serial data is not inverted '1': Serial data is inverted
[6]	LBR	Generate LIN Sync Break '0': No effect '1': Generates a LIN sync break in Modes 3 and 0 The length is selected by USARTn_ESCR:LBL[1:0] and USARTn_EFERH:LBL2. Reading this bit always returns '0'.
[5]	MS	Master Slave Mode select '0': Master mode (LIN-USART generates the serial clock) '1': Slave mode (LIN-USART receives the external serial clock from USARTn_SCK) Master or slave mode of the LIN-USART is selected in synchronous Mode 2. This bit is fixed to '0' in operation Modes 0, 1, and 3. **Note:** If slave mode is selected, the clock source must be set to external and to 'one-to-one' (i.e. USARTn_SMR:EXT = '1' and USARTn_SMR:OTO = '1').

Table 31-13. Extended Communication Control Register (USARTn_ECCR) bits

Bit Position	Bit Field Name	Bit Description
[4]	SCDE	Serial Clock Delay Enable '0': Disable clock delay '1': Enable clock delay If this bit is set, the serial output clock is delayed by half bit time. As shown in Figure 31-65, if LIN-USART operates in master Mode 2, the delay is also considered in slave mode. This bit is ignored in Modes 0, 1, and 3 in which the read value is '0'.
[3]	SSM	Start/Stop Bit Mode Enable This bit adds start and stop bits to the synchronous data format in operation Mode 2. This bit is ignored in Modes 0, 1, and 3; in these modes the read value is '0'. '0': Disable start/stop bits in synchronous Mode 2 '1': Enable start/stop bits in synchronous Mode 2
[2]	BIE	Bus Idle Interrupt Enable This bit enables an interrupt on a bus idle condition. It enables a reception interrupt (RX-IRQ) if there is neither reception nor transmission ongoing (USARTn_ECCR:RBI = '1' and USARTn_ECCR:TBI = '1') to indicate that the bus is idle. '0': Disable bus idle interrupt '1': Enable bus idle interrupt **Note:** When using the bus idle interrupt functionality, set USARTn_ESIR:AICD = '1' to prevent that the interrupt is cleared automatically when the bus is no longer idle. Do not use in Mode 2 when USARTn_ECCR:MS = '1' (Syncronous slave mode). LIN-USART cannot reliably detect the bus state in this mode.
[1]	RBI	Reception Bus Idle Flag '0': Reception activity is ongoing 1': No reception activity **Note:** RBI cannot be used under the following conditions: Syncronous Slave mode (USART_SCRn_MD='10', USARTn_ECCR:MS='1') Syncronous Master mode with non-continuous clock (USART_SCRn_MD='10', USARTn_ECCR:MS='0', USARTn_ESCR:CCO='0') During reception of LIN-sync byte in LIN-Mode (USART_SCRn_MD='11')
[0]	TBI	Transmission Bus Idle Flag '0': Transmission is ongoing '1': No transmission activity Do not use in Mode 2 when USARTn_ECCR:MS = '1' (Syncronous slave mode). LIN-USART cannot reliably detect the bus state in this mode.

31.2.12 Extended Status/Control Register (USARTn_ESCR)

This register provides several LIN functions, direct access to the SIN and SOT pin and setting for LIN-USART synchronous clock mode.

Figure 31-15. Extended Status/Control Register (USARTn_ESCR)

USARTn_ESCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
LBIE	LBD	LBL[1]	LBL[0]	SOPE	SIOP	CCO	SCES
RpWp	Rp	RpWp	RpWp	RpWp	Rp	RpWp	RpWp
0	0	0	0	0	1	0	0

Table 31-14. Extended Status/Control Register (USARTn_ESCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	LBIE	LIN Sync Break Detection Interrupt Enable (needed for Master operation) This bit enables or disables LIN sync break interrupt. '0': LIN sync break interrupt disable '1': LIN sync break interrupt enable The LIN sync break interrupt is connected to the receive interrupt. When in Mode 3, the LBD bit is set and this bit is '1'; a receive interrupt is signaled to the Interrupt Controller. This bit is fixed to '0' in operation Modes 1 and 2. **Note:** This bit is ignored in operation Mode 0
[6]	LBD	LIN Sync Break Detected Flag (needed for Master operation) '0': No LIN sync break detected '1': LIN sync break detected in operating Mode 3 It is recommended to write '0' to the USARTn_SCR:RXE before using this bit.

Table 31-14. Extended Status/Control Register (USARTn_ESCR) bits

Bit Position	Bit Field Name	Bit Description
[5:4]	LBL	LIN Sync Break Length Selection These two bits determine how many serial bit times the LIN sync break is generated by the LIN-USART. Receiving a LIN sync break is always fixed to 11-bit times. During transmission the LIN break length can be varied from 13 to 20 bit times using these two bits and USARTn_EFERH:LBL2 bit as the most significant bit. The table for various LIN break length configuration is given below. {LBL2 LBL1 LBL0}: LIN break length '000': 13 bit times '001': 14 bit times '010': 15 bit times '011': 16 bit times '100': 17 bit times '101': 18 bit times '110': 19 bit times '111': 20 bit times
[3]	SOPE	Serial Output Pin Direct Access Enable Setting this bit to '1' enables the direct write to the SOT pin. '0': Serial Output Pin Direct Access disable '1': Serial Output Pin Direct Access enable
[2]	SIOP	Serial Input Output Pin Direct Access Reading this bit always returns the actual value of the SIN pin. This is a read-only bit, writing has no effect. The value of USARTn_SOT can be controlled by USARTn_ESCSR:SIOPS and USARTn_ESCCR:SIOPC.
[1]	CCO	Continuous Clock Output Enable This bit enables a continuous serial clock at the SCK pin if the LIN-USART operates in master Mode 2 (synchronous) and the SCK pin is configured as a clock output. '0': Continuous clock output disabled '1': Continuous clock output enabled **Note:** When this bit is '1', set USARTn_ECCR:SSM to '1' to enable the Start/Stop bit in operation Mode 2..

Table 31-14. Extended Status/Control Register (USARTn_ESCR) bits

Bit Position	Bit Field Name	Bit Description
[0]	SCES	Serial Clock Edge Select This bit inverts the serial clock signal in operation Mode 2 (synchronous communication). '0': Sampling on rising clock edge (normal) '1': Sampling on falling clock edge (inverted clock) when USARTn_ECCR:SCDE is set to '0' Received data is sampled on the falling edge of the internal clock when USARTn_ECCR:SCDE = '0'. If the USARTn_ECCR:MS register is '0' (master mode) the output clock signal is also inverted. This bit is ignored in Modes 0, 1, and 3 in which the read value is '0'.

Note: The description of the interaction of USARTn_ESCR:SOPE and the USARTn_ESCSR:SIOPS/USARTn_ESCCR:SIOPC bits is given in Table 31-19.

31.2.13 Extended Communication Control Set Register (USARTn_ECCSR)

This register sets the bits of Extended Communication Control Register.

Extended Communication Control Set Register (USARTn_ECCSR)

Figure 31-16. Extended Communication Control Set Register (USARTn_ECCSR)

USARTn_ECCSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	LBRS	read0	read0	read0	BIES	read0	read0
Rp0	Rp0Wp1	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-15. Extended Communication Control Set Register (USARTn_ECCSR) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	LBRS	LBR Set '0': No effect '1': Sets the USARTn_ECCR:LBR bit Reading this bit always returns '0'.
[5:3]	read0	-
[2]	BIES	BIE Set '0': No effect '1': Sets the USARTn_ECCR:BIE bit Reading this bit always returns '0'.
[1:0]	read0	-

31.2.14 Extended Status/Control Set Register (USARTn_ESCSR)

This register sets the bits of Extended Status/Control Register.

Extended Status/Control Set Register (USARTn_ESCSR)

Figure 31-17. Extended Status/Control Set Register (USARTn_ESCSR)

USARTn_ESCSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
LBIES	read0	read0	read0	read0	SIOPS	read0	read0
Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-16. Extended Status/Control Set Register (USARTn_ESCSR) bits

Bit Position	Bit Field Name	Bit Description
[7]	LBIES	LBIE Set '0': No effect '1': Sets the USARTn_ESCR:LBIE bit Reading this bit always returns '0'.
[6:3]	read0	-
[2]	SIOPS	SOT Pin Set '0': No effect '1': Sets the SOT pin to '1' Reading this bit always returns '0'.
[1:0]	read0	-

Note: The description of the interaction of USARTn_ESCR:SOPE and the USARTn_ESCSR:SIOPS/USARTn_ESCCR:SIOPC bits is given in Table 31-19.

31.2.15 Extended Communication Control Clear Register (USARTn_ECCCR)

This register clears the bits of Extended Communication Control Register.

Extended Communication Control Clear Register (USARTn_ECCCR)

Figure 31-18. Extended Communication Control Clear Register (USARTn_ECCCR)

USARTn_ECCCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	BIEC	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-17. Extended Communication Control Clear Register (USARTn_ECCCR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	BIEC	BIE Clear '0': No effect '1': Clears the USARTn_ECCR:BIE bit Reading this bit always returns '0'.
[1:0]	read0	-

31.2.16 Extended Status/Control Clear Register (USARTn_ESCCR)

This register clears the bits of Extended Status/Control Register.

Extended Status/Control Clear Register (USARTn_ESCCR)

Figure 31-19. Extended Status/Control Clear Register (USARTn_ESCCR)

USARTn_ESCCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
LBIEC	LBDC	read0	read0	read0	SIOPC	read0	read0
Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-18. Extended Status/Control Clear Register (USARTn_ESCCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	LBIEC	LBIE Clear '0': No effect '1': Clears the USARTn_ESCR:LBIE bit Reading this bit always returns '0'.
[6]	LBDC	LBD Clear '0': No effect '1': Clears the USARTn_ESCR:LBD bit Reading this bit always returns '0'.
[5:3]	read0	-
[2]	SIOPC	SOT Pin Clear '0': No effect '1': Sets SOT pin to '0' Reading this bit always returns '0'.
[1:0]	read0	-

Note: The description of the interaction of USARTn_ESCR:SOPE and the USARTn_ESCSR:SIOPS/USARTn_ESCCR:SIOPC bits is given in Table 31-19.

Table 31-19. Description of the interaction of USARTn_ESCR:SOPE and USARTn_ESCSR:SIOPS/USARTn_ESCCR:SIOPC bits

USARTn_ESCR: SOPE	USARTn_ESCSR: SIOPS	USARTn_ESCCR: SIOPC	Function
'0'	X	X	No write happens to SOT pin
'0'	X	X	No write happens to SOT pin
'1'	'1'	X	Writes '1' to SOT pin
'1'	'0'	'1'	Writes '0' to SOT pin

31.2.17 Extended Serial Interrupt Register (USARTn_ESIR)

The Extended Serial Interrupt Register contains control bits to change the interrupt handling of the LIN-USART for improved interrupt handling and to enable DMA to handle LIN-USART data transfers and to enable the error interrupts.

Extended Serial Interrupt Register (USARTn_ESIR)

Figure 31-20. Extended Serial Interrupt Register (USARTn_ESIR)

USARTn_ESIR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	PEIE	OREIE	FREIE	TDRE	RDRF	RBI	AICD
Rp0	RpWp	RpWp	RpWp	Rp	Rp	Rp	RpWp
0	0	0	0	1	0	X	0

Table 31-20. Extended Serial Interrupt Register (USARTn_ESIR) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	PEIE	Parity Error Interrupt Enable bit '0': PE interrupt is disabled '1': PE interrupt is enabled When this bit is enabled, an error interrupt request is asserted when USARTn_SSR:PE is set.
[5]	OREIE	Overrun Error Interrupt Enable bit '0': ORE interrupt is disabled '1': ORE interrupt is enabled When this bit is enabled, an error interrupt request is asserted when USARTn_SSR:ORE is set.
[4]	FREIE	Framing Error Interrupt Enable bit. '0': FRE interrupt is disabled '1': FRE interrupt is enabled When this bit is enabled, an error interrupt request is asserted when USARTn_SSR:FRE is set.

Table 31-20. Extended Serial Interrupt Register (USARTn_ESIR) bits

Bit Position	Bit Field Name	Bit Description
[3]	TDRE	Transmission Data Register Empty (Sticky Behaviour) This flag indicates the status of the Transmission Data Register (USARTn_TDR). It has the same function as the USARTn_SSR:TDRE bit but is not cleared when data is written to USARTn_TDR. '0': Transmission Data Register is full '1': Transmission Data Register is empty When USARTn_ESIR:AICD = '0', USARTn_SSR:TDRE is used to generate a transmission interrupt. When USARTn_ESIR:AICD = '1', USARTn_ESIR:TDRE is used to generate a transmission interrupt. **Notes:** 1. This bit is set to '1' (USARTn_TDR empty) as its initial value and remains this way as long as USARTn_TDR is empty. It will only stay '0' after clearing when enough data are written to USARTn_TDR to fill the FIFO above the trigger level configured in USASRTn_TFCR:TXFLC[4:0]. 2. Refer to USARTn_SSR:TDRE bit description for more details. Writing '1' to USARTn_ESICR:TDREC clears this bit.
[2]	RDRF	Reception Data Register Full (Sticky Behaviour) This flag indicates that a new data byte has been received after the last clearing of the flag. '0': Reception Data Register is empty '1': Reception Data Register is full This flag has the same function as USARTn_SSR:RDRF but it is not cleared when data is read from the Reception Data Register (USARTn_RDR). When USARTn_ESIR:AICD = '0', USARTn_SSR:RDRF is used to generate a reception interrupt. When USARTn_ESIR:AICD = '1', USARTn_ESIR:RDRF is used to generate a reception interrupt. Writing '1' to USARTn_ESICR:RDRFC clears this bit. **Note:** This flag stays '1' as long as not all data are read from RDRn. It will only stay '0' after clearing it, whan all data are read from RDRn. Refer to USARTn_SSR:RDRF bit description for more details.

Table 31-20. Extended Serial Interrupt Register (USARTn_ESIR) bits

Bit Position	Bit Field Name	Bit Description
[1]	RBI	Reception Bus Idle (Sticky Behaviour) This flag has the same function as USARTn_ECCR:RBI but it is not cleared when the reception bus is no longer idle. '0': Reception is ongoing or finished, '1': No reception activity since last clear When USARTn_ESIR:AICD = '0' and USARTn_ECCR:BIE = '1' then USARTn_ECCR:RBI is used to generate a transmission interrupt. When USARTn_ESIR:AICD = '1' and USARTn_ECCR:BIE = '1' then USARTn_ESIR:RBI is used to generate a transmission interrupt. **Note:** When an interrupt shall be generated by a bus idle condition, set USARTn_ESIR:AICD = '1'. Because when USARTn_ESIR:AICD = '0', USARTn_ECCR:RBI would be used for interrupt generation. In case the bus is no longer idle, USARTn_ESIR:RBI is '0', and the interrupt may be removed even before the interrupt handling routine could acknowledge it. Writing '1' to USARTn_ESICR:RBIC clears this bit.
[0]	AICD	Auto Interrupt Clear Disable '0': Auto interrupt clear is enabled USARTn_SSR:TDRE, USARTn_SSR:RDR or USARTn_ECCR:RBI is used to generate interrupts. '1': Auto interrupt clear is disabled USARTn_ESIR:TDRE, USARTn_ESIR:RDRF or USARTn_ESIR:RBI is used to generate interrupts

31.2.18 Extended Interrupt Enable Register (USARTn_EIER)

This register specifies interrupt enables for various interrupt sources.

Extended Interrupt Enable Register (USARTn_EIER)

Figure 31-21. Extended Interrupt Enable Register (USARTn_EIER)

USARTn_EIER							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXFIE	RXFIE	read0	read0	read0	read0	BUSERRIE	LBSOIE
RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-21. Extended Interrupt Enable Register (USARTn_EIER) bits

Bit Position	Bit Field Name	Bit Description
[7]	TXFIE	Transmission FIFO Interrupt Enable '0': Transmission FIFO trigger level interrupt is disabled '1': Transmission FIFO trigger level interrupt is enabled **Note:** When TX FIFO is enabled (USARTn_TFCR:TXFE = '1'), a transmission interrupt is asserted if the USART-n_SSR:TDRE flag and USARTn_EIER:TXFIE are set.
[6]	RXFIE	Reception FIFO Interrupt Enable '0': Reception FIFO trigger level interrupt is disabled ' 1': Reception FIFO trigger level interrupt is enabled **Note:** When RX FIFO is enabled (USARTn_RFCR:RXFE = '1'), a Reception Interrupt is asserted if the USART-n_SSR:RDRF flag and USARTn_EIER:RXFIE are set.
[1]	BUSERRIE	Bus Error Interrupt Enable '0': Error interrupt corresponding to bus error is disabled '1': Error interrupt corresponding to bus error is enabled This bit can only be used in LIN mode and if the SOT and SIN pins are connected to the K-line adapter or the LIN-transceiver, and if reception is enabled (USARTn_SCR:RXE = '1').

Table 31-21. Extended Interrupt Enable Register (USARTn_EIER) bits

Bit Position	Bit Field Name	Bit Description
[0]	LBSOIE	Last Bit Shifted Out Interrupt Enable bit '0': Transmission interrupt corresponding to last bit shift out is disabled '1': Transmission interrupt corresponding to last bit shift out in byte field is enabled This interrupt can be used as a replacement for the Transmission Data Register empty interrupt (USARTn_SSR:TIE). Only one of these interrupts should be enabled at the same time.
[5:2]	read0	-

31.2.19 Extended Serial Interrupt Set Register (USARTn_ESISR)

This register sets the interrupt enable bits of Extended Serial Interrupt Register.

Extended Serial Interrupt Set Register (USARTn_ESISR)

Figure 31-22. Extended Serial Interrupt Set Register (USARTn_ESISR)

USARTn_ESISR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	PEIES	OREIES	FREIES	read0	read0	read0	read0
Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-22. Extended Serial Interrupt Set Register (USARTn_ESISR) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	PEIES	Parity Error (in data received) Interrupt Enable Set '0': No effect '1': Sets USARTn_ESIR:PEIE to '1' Reading this bit always returns '0'.
[5]	OREIES	Overrun Error Interrupt Enable Set '0': No effect '1': Sets USARTn_ESIR:OREIE to '1' Reading this bit always returns '0'.
[4]	FREIES	Framing Error Interrupt Enable Set '0': No effect '1': Sets USARTn_ESIR:FREIE to '1' Reading this bit always returns '0'.
[3:0]	read0	-

31.2.20 Extended Interrupt Enable Set Register (USARTn_EIESR)

This register sets the bits of Extended Interrupt Enable Register.

Extended Interrupt Enable Set Register (USARTn_EIESR)

Figure 31-23. Extended Interrupt Enable Set Register (USARTn_EIESR)

USARTn_EIESR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXFIES	RXFIES	read0	read0	read0	read0	BUSERRIES	LBSOIES
Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-23. Extended Interrupt Enable Set Register (USARTn_EIESR) bits

Bit Position	Bit Field Name	Bit Description
[7]	TXFIES	Transmission FIFO Interrupt Enable Set '0': No effect '1': Sets USARTn_EIER:TXFIE to '1' Reading this bit always returns '0'.
[6]	RXFIES	Reception FIFO Interrupt Enable Set '0': No effect '1': Sets USARTn_EIER:RXFIE to '1' Reading this bit always returns '0'.
[5:2]	read0	-
[1]	BUSERRIES	Bus Error Interrupt Enable Set '0': No effect '1': Sets USARTn_EIER:BUSERRIE to '1' Reading this bit always returns '0'.
[0]	LBSOIES	Last Bit Shifted Out Interrupt Enable Set '0': No effect '1': Sets USARTn_EIER:LBSOIE to '1' Reading this bit always returns '0'.

31.2.21 Extended Serial Interrupt Clear Register (USARTn_ESICR)

This register clears the bits of Extended Serial Interrupt Register.

Extended Serial Interrupt Clear Register (USARTn_ESICR)

Figure 31-24. Extended Serial Interrupt Clear Register (USARTn_ESICR)

USARTn_ESICR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	PEIEC	OREIEC	FREIEC	TDREC	RDRFC	RBIC	read0
Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0
0	0	0	0	0	0	0	0

Table 31-24. Extended Serial Interrupt Clear Register (USARTn_ESICR) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	PEIEC	Parity Error Interrupt Enable Clear '0': No effect '1': Clears USARTn_ESIR:PEIE Reading this bit always returns '0'.
[5]	OREIEC	Overrun Error Interrupt Enable Clear '0': No effect '1': Clears USARTn_ESIR:OREIE Reading this bit always returns '0'.
[4]	FREIEC	Framing Error Interrupt Enable Clear '0': No effect '1': Clears USARTn_ESIR:FREIE Reading this bit always returns '0'.
[3]	TDREC	Transmission Data Register Empty Flag Clear '0': No effect '1': Clears USARTn_ESIR:TDRE Reading this bit always returns '0'.
[2]	RDRFC	Reception Data Register Full Flag Clear '0': No effect '1': Clears USARTn_ESIR:RDRF Reading this bit always returns '0'.

Table 31-24. Extended Serial Interrupt Clear Register (USARTn_ESICR) bits

Bit Position	Bit Field Name	Bit Description
[1]	RBIC	Reception Bus Idle Flag Clear '0': No effect '1': Clears USARTn_ESIR:RBI Reading this bit always returns '0'.
[0]	read0	-

31.2.22 Extended Interrupt Enable Clear Register (USARTn_EIECR)

This register clears the bits of Extended Interrupt Enable Register.

Extended Interrupt Enable Clear Register (USARTn_EIECR)

Figure 31-25. Extended Interrupt Enable Clear Register (USARTn_EIECR)

USARTn_EIECR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXFIEC	RXFIEC	read0	read0	read0	read0	BUSERRIEC	LBSOIEC
Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-25. Extended Interrupt Enable Clear Register (USARTn_EIECR) bits

Bit Position	Bit Field Name	Bit Description
[7]	TXFIEC	Transmission FIFO Interrupt Enable Clear '0': No effect '1': Clears USARTn_EIER:TXFIE Reading this bit always returns '0'.
[6]	RXFIEC	Reception FIFO Interrupt Enable Clear '0': No effect '1': Clears USARTn_EIER:RXFIE Reading this bit always returns '0'.
[5:2]	read0	-
[1]	BUSERRIEC	Bus Error Interrupt Enable Clear '0': No effect '1': Clears USARTn_EIER:BUSERRIE Reading this bit always returns '0'.
[0]	LBSOIEC	Last Bit Shifted Out Interrupt Enable Clear '0': No effect '1': Clears USARTn_EIER:LBSOIE Reading this bit always returns '0'.

31.2.23 Extended Feature Enable Register - L (USARTn_EFERL)

This register enables the transmission or reception a LIN-frame header. It also disables the resetting of the reception state machine when USARTn_SCR:CRE is set and disables the detection of the low level of SIN after FRE as a start bit, interrupt loop-back testing, detection of a bus error and the most significant bit of LIN break length.

Extended Feature Enable Register - L (USARTn_EFERL)

Figure 31-26. Extended Feature Enable Register - L (USARTn_EFERL)

USARTn_EFERL							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	OSDE	DTSTART	RSTRFM	LBEDGE	read0	read0	read0
Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-26. Extended Feature Enable Register - L (USARTn_EFERL) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	OSDE	Oversampling Disable '0': Five times oversampling of asynchronous serial data input is enabled '1': Five times oversampling of asynchronous serial data input is disabled
[5]	DTSTART	Detect Start bit '0': Low level on SIN after a framing error (USARTn_SSR:FRE) is detected as the start bit '1': Next falling edge of SIN after a framing error (USARTn_SSR:FRE) is detected as the start bit **Note:** Always set this bit to '0' in Mode 2 (USARTn_SCR:MD[1:0]='10')."
[4]	RSTRFM	Reset Reception State Machine '0': Resetting of reception state machine is enabled when USARTn_SCR:CRE is set '1': Resetting of reception state machine is disabled when USARTn_SCR:CRE is set

Table 31-26. Extended Feature Enable Register - L (USARTn_EFERL) bits

Bit Position	Bit Field Name	Bit Description
[3]	LBEDGE	LIN Break Edge Sensitive '0': LIN break is level sensitive '1': LIN break is edge sensitive When this bit is '0', the LIN-USART, in operation Mode 3 (LIN-mode), detects a low level on SIN, which indicates the start of a LIN break. When this bit is '1', the LIN-USART, in operation Mode 3 (LIN-mode), detects a falling edge on SIN, which indicates the start of a LIN break.
[2:0]	read0	-

31.2.24 Extended Feature Enable Register - H (USARTn_EFERH)

This register enables internal loop-back testing, Frame-ID register, detection of bus error, and MSB of LIN break length.

Extended Feature Enable Register - H (USARTn_EFERH)

Figure 31-27. Extended Feature Enable Register - H (USARTn_EFERH)

USARTn_EFERH							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	INTLBEN	BRGR	read0	DBE	read0	LBL2
Rp0	Rp0	RpWp	RpWp	Rp0	RpWp	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 31-27. Extended Feature Enable Register - H (USARTn_EFERH) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5]	INTLBEN	Internal Loop Back Enable '0': Internal loop back is disabled '1': Internal loop back is enabled When the internal loop back is enabled, the USARTn_SOT output is internally fed back to the internal USARTn_SIN input so that software can compare the transmitted and received value. If USARTn_EFERH:INTLBEN is set, USARTn_SCR:RXE is set automatically and SOT always outputs '1'.
[4]	BRGR	Baud Rate Register Read '0': A read of the Baud Rate Generation Register (USARTn_BGR) returns the current reload counter value '1': A read of Baud Rate Generation Register (USARTn_BGR) returns the reload value written by the CPU
[3]	read0	-

Table 31-27. Extended Feature Enable Register - H (USARTn_EFERH) bits

Bit Position	Bit Field Name	Bit Description
[2]	DBE	Detection of Bus Error Enable '0': Detection of bus error is disabled '1': Detection of bus error is enabled **Notes:** 1. This feature can only be used in operation Mode 3 (LIN mode), or if SOT and SIN are connected to the K-line adapter and if reception is enabled (USARTn_SCR:RXE = '1'). 2. Enable/Disable of Bus Error Detect feature should only be done when the bus is idle. 3. The bus error will be continuously triggered if bus error detection is enabled (USARTn_EFERH:DBE = '1') in internal loop back mode (USARTn_EFERH:INTLBEN = '1') because in internal loop back mode, SOT is always '1'.
[1]	read0	-
[0]	LBL2	MSB of LIN Break Length '0': USARTn_ESCR:LBL[1:0] sets LIN break length from 13 to 16 bit times '1': USARTn_ESCR:LBL[1:0] sets LIN break length from 17 to 20 bit times

31.2.25 Reception FIFO Control Register (USARTn_RFCR)

This register specifies interrupt level of reception FIFO, enables the reception FIFO, and resets the FIFO pointer.

Reception FIFO Control Register (USARTn_RFCR)

Figure 31-28. Reception FIFO Control Register (USARTn_RFCR)

USARTn_RFCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RXFE	RXFCL	read0	RXFLC[4]	RXFLC[3]	RXFLC[2]	RXFLC[1]	RXFLC[0]
RpWp	Rp0Wp1	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-28. Reception FIFO Control Register (USARTn_RFCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	RXFE	Reception FIFO enable '0': Reception FIFO is disabled '1': Reception FIFO is enabled
[6]	RXFCL	Reception FIFO clear '0': No effect '1': Reception FIFO pointer is reset to '00000' **Note:** Reception FIFO pointers are also cleared when a mode change occurs or when the USARTn_SMR:UPCL register bit is set to '1'. Reading this bit always returns 0.
[5]	read0	-

Table 31-28. Reception FIFO Control Register (USARTn_RFCR) bits

Bit Position	Bit Field Name	Bit Description
[4:0]	RXFLC	Reception FIFO Level Configuration These bits specify the reception FIFO interrupt level. When the number of data in the reception FIFO is greater than or equal to the trigger level value programmed by these bits a reception interrupt is asserted. Reading these bits returns the written value. '00000':Not allowed '00001':RX interrupt if number of data is greater than or equal to 1 data byte '00010':RX interrupt if number of data is greater than or equal to 2 data bytes '00011':RX interrupt if number of data is greater than or equal to 3 data bytes '00100':RX interrupt if number of data is greater than or equal to 4 data bytes '00101':RX interrupt if number of data is greater than or equal to 5 data bytes '00110':RX interrupt if number of data is greater than or equal to 6 data bytes '00111':RX interrupt if number of data is greater than or equal to 7 data bytes '01000':RX interrupt if number of data is greater than or equal to 8 data bytes '01001':RX interrupt if number of data is greater than or equal to 9 data bytes '01010':RX interrupt if number of data is greater than or equal to 10 data bytes '01011':RX interrupt if number of data is greater than or equal to 11 data bytes '01100':RX interrupt if number of data is greater than or equal to 12 data bytes '01101':RX interrupt if number of data is greater than or equal to 13 data bytes '01110':RX interrupt if number of data is greater than or equal to 14 data bytes '01111':RX interrupt if number of data is greater than or equal to 15 data bytes '10000':RX interrupt if number of data is greater than or equal to 16 data bytes **Note:** The bit combination from 17 to 31 are not allowed as the FIFO size is 16 bytes.

31.2.26 Transmission FIFO Control Register (USARTn_TFCR)

This register specifies the interrupt level for transmission FIFO, enables the transmission FIFO, and resets the FIFO pointer.

Transmission FIFO Control Register (USARTn_TFCR)

Figure 31-29. Transmission FIFO Control Register (USARTn_TFCR)

USARTn_TFCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXFE	TXFCL	read0	TXFLC[4]	TXFLC[3]	TXFLC[2]	TXFLC[1]	TXFLC[0]
RpWp	Rp0Wp1	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-29. Transmission FIFO Control Register (USARTn_TFCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	TXFE	Transmission FIFO Enable '0': Transmission FIFO is disabled '1': Transmission FIFO is enabled
[6]	TXFCL	Transmission FIFO Clear '0': No effect '1': Transmission FIFO pointer is reset to '00000' **Note:** Transmission FIFO pointers are also cleared when a mode change occurs or when the USARTn_SMR:UPCL register bit is set to '1'. Reading this bit always returns '0'.
[5]	read0	-

Table 31-29. Transmission FIFO Control Register (USARTn_TFCR) bits

Bit Position	Bit Field Name	Bit Description
[4:0]	TXFLC	Transmission FIFO Interrupt Level Configuration These bits specify the transmission FIFO interrupt level. When the number of data in the transmission FIFO is less or equal to the trigger level programmed by these bits a transmission interrupt is asserted. Reading these bits returns the written value. '00000':TX interrupt if TX FIFO is empty '00001':TX interrupt if number of data is less than or equal to 1 data byte '00010':TX interrupt if number of data is less than or equal to 2 data bytes '00011':TX interrupt if number of data is less than or equal to 3 data bytes '00100':TX interrupt if number of data is less than or equal to 4 data bytes '00101':TX interrupt if number of data is less than or equal to 5 data bytes '00110':TX interrupt if number of data is less than or equal to 6 data bytes '00111':TX interrupt if number of data is less than or equal to 7 data bytes '01000':TX interrupt if number of data is less than or equal to 8 data bytes '01001':TX interrupt if number of data is less than or equal to 9 data bytes '01010':TX interrupt if number of data is less than or equal to 10 data bytes '01011':TX interrupt if number of data is less than or equal to 11 data bytes '01100':TX interrupt if number of data is less than or equal to 12 data bytes '01101':TX interrupt if number of data is less than or equal to 13 data bytes '01110':TX interrupt if number of data is less than or equal to 14 data bytes '01111':TX interrupt if number of data is less than or equal to 15 data bytes **Note:** The bit combination from 16 to 31 are not allowed as the FIFO size is 16 bytes.

31.2.27 Reception FIFO Control Set Register (USARTn_RFCSR)

This register sets the bits of Reception FIFO Control Register.

Reception FIFO Control Set Register (USARTn_RFCSR)

Figure 31-30. Reception FIFO Control Set Register (USARTn_RFCSR)

USARTn_RFCSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RXFES	read0	read0	read0	read0	read0	read0	read0
Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-30. Reception FIFO Control Set Register (USARTn_RFCSR) bits

Bit Position	Bit Field Name	Bit Description
[7]	RXFES	RXFE Set '0': No effect '1': Sets the USARTn_RFCR:RXFE bit Reading this bit always returns '0'.
[6:0]	read0	-

31.2.28 Transmission FIFO Control Set Register (USARTn_TFCSR)

This register sets the bits of Transmission FIFO Control Register

Figure 31-31. Transmission FIFO Control Set Register (USARTn_TFCSR)

USARTn_TFCSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXFES	read0	read0	read0	read0	read0	read0	read0
Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-31. Transmission FIFO Control Set Register (USARTn_TFCSR) bits

Bit Position	Bit Field Name	Bit Description
[7]	TXFES	TXFE Set '0': No effect '1': Sets the USARTn_TFCR:TXFE bit Reading this bit always returns '0'.
[6:0]	read0	-

31.2.29 Reception FIFO Control Clear Register (USARTn_RFCCR)

This register clears the bits of Reception FIFO Control Register.

Reception FIFO Control Clear Register (USARTn_RFCCR)

Figure 31-32. Reception FIFO Control Clear Register (USARTn_RFCCR)

USARTn_RFCCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RXFEC	read0	read0	read0	read0	read0	read0	read0
Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-32. Reception FIFO Control Clear Register (USARTn_RFCCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	RXFEC	RXFE Clear '0': No effect '1': Clears the USARTn_RFCR:RXFE bit Reading this bit always returns '0'.
[6:0]	read0	-

31.2.30 Transmission FIFO Control Clear Register (USARTn_TFCCR)

This register clears the bits of Transmission FIFO Control Register.

Transmission FIFO Control Clear Register (USARTn_TFCCR)

Figure 31-33. Transmission FIFO Control Clear Register (USARTn_TFCCR)

USARTn_TFCCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TXFEC	read0	read0	read0	read0	read0	read0	read0
Rp0Wp1	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-33. Transmission FIFO Control Clear Register (USARTn_TFCCR) bits

Bit Position	Bit Field Name	Bit Description
[7]	TXFEC	TXFE Clear '0': No effect '1': Clears the USARTn_TFCR:TXFE bit Reading this bit always returns '0'.
[6:0]	read0	-

31.2.31 Reception FIFO Status Register (USARTn_RFSR)

The Reception FIFO Status Register indicates the number of valid data available in the reception FIFO.

Reception FIFO Status Register (USARTn_RFSR)

Figure 31-34. Reception FIFO Status Register (USARTn_RFSR)

USARTn_RFSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	RXFVD[4]	RXFVD[3]	RXFVD[2]	RXFVD[1]	RXFVD[0]
Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 31-34. Reception FIFO Status Register (USARTn_RFSR) bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	RXFVD	Reception FIFO Valid Data These bits indicate the number of valid data in the reception FIFO. '00000':Reception FIFO contains no valid data byte '00001':Reception FIFO contains 1 valid data byte '00010':Reception FIFO contains 2 valid data bytes '00011':Reception FIFO contains 3 valid data bytes '00100':Reception FIFO contains 4 valid data bytes '00101':Reception FIFO contains 5 valid data bytes '00110':Reception FIFO contains 6 valid data bytes '00111':Reception FIFO contains 7 valid data bytes '01000':Reception FIFO contains 8 valid data bytes '01001':Reception FIFO contains 9 valid data bytes '01010':Reception FIFO contains 10 valid data bytes '01011':Reception FIFO contains 11 valid data bytes '01100':Reception FIFO contains 12 valid data bytes '01101':Reception FIFO contains 13 valid data bytes '01110':Reception FIFO contains 14 valid data bytes '01111':Reception FIFO contains 15 valid data bytes '10000':Reception FIFO contains 16 valid data bytes **Note:** The bit combination from 17 to 31 are not used as the FIFO size is 16 bytes.

31.2.32 Transmission FIFO Status Register (USARTn_TFSR)

This register indicates the number of valid data bytes available in the transmit FIFO.

Transmission FIFO Status Register (USARTn_TFSR)

Figure 31-35. Transmission FIFO Status Register (USARTn_TFSR)

USARTn_TFSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	TXFVD[4]	TXFVD[3]	TXFVD[2]	TXFVD[1]	TXFVD[0]
Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 31-35. Transmission FIFO Status Register (USARTn_TFSR) bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	TXFVD	Transmission FIFO Valid Data These bits indicates the number of valid data in the transmission FIFO. '00000':Transmission FIFO contains no valid data byte '00001':Transmission FIFO contains 1 valid data byte '00010':Transmission FIFO contains 2 valid data bytes '00011':Transmission FIFO contains 3 valid data bytes '00100':Transmission FIFO contains 4 valid data bytes '00101':Transmission FIFO contains 5 valid data bytes '00110':Transmission FIFO contains 6 valid data bytes '00111':Transmission FIFO contains 7 valid data bytes '01000':Transmission FIFO contains 8 valid data bytes '01001':Transmission FIFO contains 9 valid data bytes '01010':Transmission FIFO contains 10 valid data bytes '01011':Transmission FIFO contains 11 valid data bytes '01100':Transmission FIFO contains 12 valid data bytes '01101':Transmission FIFO contains 13 valid data bytes '01110':Transmission FIFO contains 14 valid data bytes '01111':Transmission FIFO contains 15 valid data bytes **Note:** The bit combination from 16 to 31 are not used as the FIFO size is 16 bytes.

31.2.33 Extended Status Register (USARTn_ESR)

This register indicates the status of completion of the last bit of a byte and bus error detection. Additionally the USARTn_ESR:AD bit of the multiprocessor Mode 1 can be read.

Extended Status Register (USARTn_ESR)

Figure 31-36. Extended Status Register (USARTn_ESR)

USARTn_ESR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	AD	read0	read0	LBSOF	BUSERR	read0	read0
Rp0	Rp	Rp0	Rp0	Rp	Rp	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-36. Extended Status Register (USARTn_ESR) bits

Bit Position	Bit Field Name	Bit Description
[7]	read0	-
[6]	AD	AD Status of Received Frame This bit specifies the AD status of the last received frame in operation Mode 1. '0': Indicates a normal data frame '1': Indicates an address frame This is a read-only bit and writing has no effect.
[5:4]	read0	-

Table 31-36. Extended Status Register (USARTn_ESR) bits

Bit Position	Bit Field Name	Bit Description
[3]	LBSOF	Last Bit Shift Out Flag '0': Last bit of transmission data byte not yet shifted out '1': Last bit of transmission data byte shifted out This bit is set when the last data bit of the data byte or the last stop bit was shifted out if one of the two following conditions occur: In synchronous mode (Mode 2) and if LIN-USART is master (i.e. USARTn_ECCR:MS = '0') Or the continuous clock mode is enabled in slave mode (i.e. USARTn_ECCR:MS='1' and USARTn_ESCR:CCO='1'). In case of synchronous mode (Mode 2) and continuous clock is disabled in slave mode (i.e. USARTn_ECCR:MS = '1' and USARTn_ESCR:CCO = '0'), this flag is set in the middle of the last data bit or stop bit. An transmission interrupt is asserted if USARTn_EIER:BSOIE is '1'. In case of asynchronous mode, this bit is set when the stop bit was shifted out. This bit is cleared by writing '1' to USARTn_ESCLR:LBSOFC bit or by writing data to USARTn_TDR/TX FIFO.
[2]	BUSERR	Bus Error Flag '0': No physical bus error detected '1': Physical bus error detected This bit can only be used in LIN mode and if SOT and SIN are connected to K-line adapter or LIN-Tranceiver. This bit is cleared by writing '1' to USARTn_ESCLR:BUSERRC bit. An error interrupt is asserted if USARTn_EIER:BUSERRIE is '1'.
[1:0]	read0	-

31.2.34 Extended Status Clear Register (USARTn_ESCLR)

This register clears the bits of Extended Status Register.

Extended Status Clear Register (USARTn_ESCLR)

Figure 31-37. Extended Status Clear Register (USARTn_ESCLR)

USARTn_ESCLR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	LBSOFC	BUSERRC	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 31-37. Extended Status Clear Register (USARTn_ESCLR) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3]	LBSOFC	LBSOF Clear '0': No effect '1': Clears the USARTn_ESR:LBSOF bit Reading this bit always returns '0'.
[2]	BUSERRC	BUSERR Clear '0': No effect '1': Clears the USARTn_ESR:BUSERR bit Reading this bit always returns '0'.
[1:0]	read0	-

31.2.35 Baud Rate Generation Reload Register (USARTn_BGRLL)

The Baud Rate/Reload Counter Register sets the division ratio for the serial clock.

Baud Rate Generation Reload Register - L (USARTn_BGRLL)

Figure 31-38. Baud Rate Generation Reload Register - L (USARTn_BGRLL)

USARTn_BGRLL							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
VALUE[7]	VALUE[6]	VALUE[5]	VALUE[4]	VALUE[3]	VALUE[2]	VALUE[1]	VALUE[0]
Wp	Wp	Wp	Wp	Wp	Wp	Wp	Wp
0	0	0	0	0	0	0	0

Table 31-38. Baud Rate Generation Reload Register - L (USARTn_BGRLL) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	VALUE	Baud Rate Generation Reload Write Register See 31.7 LIN-USART baud rates which explains how to calculate the reload value from the baud rate. The lower 8 bits of the 19-bit reload value must be written to this register

31.2.36 Baud Rate Generation Reload Register(USARTn_BGRLM)

The Baud Rate/Reload Counter Register sets the division ratio for the serial clock.

Baud Rate Generation Reload Register - M (USARTn_BGRLM)

Figure 31-39. Baud Rate Generation Reload Register - M (USARTn_BGRLM)

USARTn_BGRLM							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
VALUE[7]	VALUE[6]	VALUE[5]	VALUE[4]	VALUE[3]	VALUE[2]	VALUE[1]	VALUE[0]
Wp	Wp	Wp	Wp	Wp	Wp	Wp	Wp
0	0	0	0	0	0	0	0

Table 31-39. Baud Rate Generation Reload Register - M (USARTn_BGRLM) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	VALUE	Baud Rate Generation Reload Write Register See 31.7 LIN-USART baud rates which explains how to calculate the reload value from the baud rate. The middle 8 bits of the 19-bit reload value must be written to this register.

31.2.37 Baud Rate Generation Reload Register (USARTn_BGRLH)

The Baud Rate/Reload Counter Register sets the division ratio for the serial clock.

Baud Rate Generation Reload Register - H (USARTn_BGRLH)

Figure 31-40. Baud Rate Generation Reload Register - H (USARTn_BGRLH)

USARTn_BGRLH							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	VALUE[2]	VALUE[1]	VALUE[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Wp	Wp	Wp
0	0	0	0	0	0	0	0

Table 31-40. Baud Rate Generation Reload Register - H (USARTn_BGRLH) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2:0]	VALUE	Baud Rate Generation Reload Write Register See 31.7 LIN-USART baud rates which for explains how to calculate the reload value from the baud rate. The upper 3 bits of the 19-bit reload value must be written to this register.

The Baud Rate Generation Reload Register (USARTn_BGRL) determines the division ratio for the serial clock and it is a write-only register. It can be written via 8-, 16- and 32-bit access. If the automatic baud rate detection is enabled, the value written to the baud rate generation counter USARTn_BGRL is overwritten when the baud rate is detected by LIN-USART.

31.2.38 Baud Rate Generation Register (USARTn_BGRL)

The current counter value of the transmission reload counter or the reload value can be read by reading this register.

Baud Rate Generation Register - L (USARTn_BGRL)

Figure 31-41. Baud Rate Generation Register - L (USARTn_BGRL)

USARTn_BGRL							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BGRVALUE[7]	BGRVALUE[6]	BGRVALUE[5]	BGRVALUE[4]	BGRVALUE[3]	BGRVALUE[2]	BGRVALUE[1]	BGRVALUE[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 31-41. Baud Rate Generation Register - L (USARTn_BGRL)

Bit Position	Bit Field Name	Bit Description
[7:0]	BGRVALUE	Baud Rate Generation Read Register When read, this register indicates the lower 8 bits of the 19-bit reload counter value. This register is a read-only register. The read value is controlled by USARTn_EFERH:BRGR bit.

31.2.39 Baud Rate Generation Register (USARTn_BGRM)

The current counter value of the transmission reload counter or the reload value can be read by reading this register.

Baud Rate Generation Register - M (USARTn_BGRM)

Figure 31-42. Baud Rate Generation Register - M (USARTn_BGRM)

USARTn_BGRM							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BGRVALUE[7]	BGRVALUE[6]	BGRVALUE[5]	BGRVALUE[4]	BGRVALUE[3]	BGRVALUE[2]	BGRVALUE[1]	BGRVALUE[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 31-42. Baud Rate Generation Register - M (USARTn_BGRM) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	BGRVALUE	Baud Rate Generation Read Register When read, this register indicates the middle 8 bits of 19-bit reload counter value. This register is a read-only register. Read value is controlled by USARTn_EFERH:BRGR bit.

31.2.40 Baud Rate Generation Register (USARTn_BGRH)

The current counter value of the transmission reload counter or the reload value can be read by reading this register.

Baud Rate Generation Register - H (USARTn_BGRH)

Figure 31-43. Baud Rate Generation Register - H (USARTn_BGRH)

USARTn_BGRH							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	BGRVALUE[2]	BGRVALUE[1]	BGRVALUE[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 31-43. Baud Rate Generation Register - H (USARTn_BGRH) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2:0]	BGRVALUE	Baud Rate Generation Read Register When read, this register indicates the upper 3 bits of 19-bit reload counter value. This register is a read-only register. Read value is controlled by USARTn_EFERH:BRGR bit.

USARTn:BGR register can only be read via 32-bit read access.

31.2.41 Serial Transmit DMA Configuration Register (USARTn_STXDR)

This register configures the control bits for transmit DMA function.

Serial Transmit DMA Configuration Register (USARTn_STXDR)

Figure 31-44. Serial Transmit DMA Configuration Register (USARTn_STXDR)

USARTn_STXDR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	TXDISDOERR	TXDRQEN	TXDDEN
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-44. Serial Transmit DMA Configuration Register (USARTn_STXDR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	TXDISDOERR	TX-DMA Request Mask/Disable '0': TX-DMA request is not masked/disabled when a transmission error (BUSERR) occurs, '1': TX-DMA request is disabled when a transmission error (BUSERR) occurs
[1]	TXDRQEN	TX-DMA Request Enable '0': TX-DMA request is disabled, '1': TX-DMA request is enabled,
[0]	TXDDEN	TX-DMA Demand Transfer Enable '0': TX-DMA demand transfer is disabled, '1': TX-DMA demand transfer is enabled When TX-DMA request is enabled and this bit is '1', demand transfer is done by DMA.

31.2.42 Serial Receive DMA Configuration Register (USARTn_SRXDR)

This register configures the control bits for receive DMA function.

Serial Receive DMA Configuration Register (USARTn_SRXDR)

Figure 31-45. Serial Receive DMA Configuration Register (USARTn_SRXDR)

USARTn_SRXDR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	RXDISDOERR	RXDRQEN	RXDDEN
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-45. Serial Receive DMA Configuration Register (USARTn_SRXDR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	RXDISDOERR	RX-DMA Request Mask/Disable '0': RX-DMA request is not masked/disabled when any of the reception errors (PE, FRE, or ORE) occurs, '1': RX-DMA request is disabled when any of the reception errors (PE, FRE, or ORE) occurs
[1]	RXDRQEN	RX-DMA Request Enable '0': RX-DMA request is disabled, '1': RX-DMA request is enabled,
[0]	RXDDEN	RX-DMA Demand Transfer Enable '0': RX-DMA demand transfer is disabled, '1': RX-DMA demand transfer is enabled When RX-DMA request is enabled and this bit is '1', demand transfer is done by DMA.

31.2.43 Serial Transmit DMA Configuration Set Register (USARTn_STXDSR)

This register sets the bits of Serial Transmit DMA Configuration Register.

Serial Transmit DMA Configuration Set Register (USARTn_STXDSR)

Figure 31-46. Serial Transmit DMA Configuration Set Register (USARTn_STXDSR)

USARTn_STXDSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	TXDISDOERRS	TXDRQENS	TXDDENS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-46. Serial Transmit DMA Configuration Set Register (USARTn_STXDSR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	TXDISDOERRS	TX-DMA Request Mask/Disable Set '0': No effect '1': Sets the TX-DMA request mask/disable bit (USART-n_STXDR:TXDISDOERR) Reading this bit always returns '0'.
[1]	TXDRQENS	TX-DMA Request Enable Set '0': No effect '1': Sets the TX-DMA request enable bit (USART-n_STXDR:TXDRQEN) Reading this bit always returns '0'.
[0]	TXDDENS	TX-DMA Demand Transfer Enable Set '0': No effect '1': Sets the TX-DMA demand transfer enable bit (USART-n_STXDR:TXDDEN) Reading this bit always returns '0'.

31.2.44 Serial Receive DMA Configuration Set Register (USARTn_SRXDSR)

This register sets the bits of Receive DMA Configuration Register.

Serial Receive DMA Configuration Set Register (USARTn_SRXDSR)

Figure 31-47. Serial Receive DMA Configuration Set Register (USARTn_SRXDSR)

USARTn_SRXDSR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	RXDISDOERRS	RXDRQENS	RXDDENS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-47. Serial Receive DMA Configuration Set Register (USARTn_SRXDSR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	RXDISDOERRS	RX-DMA Request Mask/Disable Set '0': No effect '1': Sets the RX-DMA request mask/disable bit (USARTn_S-RXDR:RXDISDOERR) Reading this bit always returns '0'.
[1]	RXDRQENS	RX-DMA Request Enable Set '0': No effect '1': Sets the RX-DMA request enable bit (USARTn_S-RXDR:RXDRQEN) Reading this bit always returns '0'.
[0]	RXDDENS	RX-DMA Demand Transfer Enable Set '0': No effect '1': Sets the RX-DMA demand transfer enable bit (USARTn_S-RXDR:RXDDEN) Reading this bit always returns '0'.

31.2.45 Serial Transmit DMA Configuration Clear Register (USARTn_STXDCR)

This register clears the bits of Transmit DMA Configuration Register.

Serial Transmit DMA Configuration Clear Register (USARTn_STXDCR)

Figure 31-48. Serial Transmit DMA Configuration Clear Register (USARTn_STXDCR)

USARTn_STXDCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	TXDISDOERRC	TXDRQENC	TXDDENC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-48. Serial Transmit DMA Configuration Clear Register (USARTn_STXDCR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	TXDISDOERRC	TX-DMA Request Mask/Disable Clear '0': No effect '1': Clears the TX-DMA request mask/disable bit (USART-n_STXDR:TXDISDOERR) Reading this bit always returns '0'.
[1]	TXDRQENC	TX-DMA Request Enable Clear '0': No effect '1': Clears the TX-DMA request enable bit (USART-n_STXDR:TXDRQEN) Reading this bit always returns '0'.
[0]	TXDDENC	TX-DMA Demand Transfer Enable Clear '0': No effect '1': Clears the TX-DMA demand transfer enable bit (USART-n_STXDR:TXDDEN) Reading this bit always returns '0'.

31.2.46 Serial Receive DMA Configuration Clear Register (USARTn_SRXDCR)

This register clears the bits of Receive DMA Configuration Register.

Serial Receive DMA Configuration Clear Register (USARTn_SRXDCR)

Figure 31-49. Serial Receive DMA Configuration Clear Register (USARTn_SRXDCR)

USARTn_SRXDCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	RXDISDOERRC	RXDRQENC	RXDDENC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 31-49. Serial Receive DMA Configuration Clear Register (USARTn_SRXDCR) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	RXDISDOERRC	RX-DMA Request Mask/Disable Clear '0': No effect '1': Clears the RX-DMA request mask/disable bit (USARTn_S-RXDR:RXDISDOERR) Reading this bit always returns '0'.
[1]	RXDRQENC	RX-DMA Request Enable Clear '0': No effect '1': Clears the RX-DMA request enable bit (USARTn_S-RXDR:RXDRQEN) Reading this bit always returns '0'.
[0]	RXDDENC	RX-DMA Demand Transfer Enable Clear '0': No effect '1': Clears the RX-DMA demand transfer enable bit (USARTn_S-RXDR:RXDDEN) Reading this bit always returns '0'.

31.2.47 Debug Register (USARTn_DEBUG)

This register enables the debug mode.

Debug Register (USARTn_DEBUG)

Figure 31-50. Debug Register (USARTn_DEBUG)

USARTn_DEBUG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	DBGEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 31-50. Debug Register (USARTn_DEBUG) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	DBGEN	Debug Enable '0': No impact of debug mode (DEBUG input) on the behaviour of LIN-USART '1': LIN-USART enters debug mode when the DEBUG input is high

Note: For the behaviour of LIN-USART in debug mode, refer to 31.9 Notes on using the LIN-USART.

31.2.48 Unused Registers

The following is information concerning unused registers.

There are several unused registers and these should be configured as shown below

Figure 31-51. Unused Register example

USARTn_Unused							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 31-51. Unused Register bits

Bit Position	Bit Field Name	Bit Description
[7:0]	read0	Always write to '0000'

31.3 Configuration of the LIN-USART

This section provides a short overview of the operation modes of the LIN-USART.

LIN-USART operation modes

The LIN-USART operates in four different modes, which are determined by USARTn_SMR:MD[0] and USARTn_SMR:MD[1] bits. Mode 0 and 2 are used for bidirectional serial communication, mode 1 for master/slave communication, and mode 3 for LIN master/slave communication.

Table 31-52. LIN-USART operation modes

Operation mode		Data length		Synchronization of mode	Length of stop bit	Data bit direction *1
		Parity disabled	Parity enabled			
0	Normal mode	7 or 8		Asynchronous	1 or 2	L/M
1	Multipro-cessor	7 or 8 + 1 *2	--	Asynchronous	1 or 2	L/M
2	Normal mode	8 or 9 (if USARTn_ECCR:SSM = '1')		Synchronous	0, 1 or 2	L/M
3	LIN mode	8	--	Asynchronous	1	L

*1: means the data bit transfer format: LSB or MSB first.

*2: '+1' means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

Note:

Mode 1 operation is supported both for master or slave operation of the LIN-USART in a master- slave connection system. In mode 3 the LIN-USART function is locked to 8N1-format, LSB first.

If the mode is changed, LIN-USART cuts off all possible transmission or reception and awaits then new action. It is strongly recommended to reset the USART (write '1' to USARTn_SMR:UPCL) after changing the operation mode.

USARTn_SMR:MD[1] and USARTn_SMR:MD[0] bits determine the operation mode of the LIN-USART as shown in the following table.

Table 31-53. Mode bit setting

USARTn_SMR:MD[1]	USARTn_SMR:MD[0]	Mode	Description
'0'	'0'	0	Asynchronous (normal mode)
'0'	'1'	1	Asynchronous (multiprocessor mode)
'1'	'0'	2	Synchronous (normal mode)
'1'	'1'	3	Asynchronous (LIN mode)

31.4 LIN-USART pins

This section describes the LIN-USART pins.

LIN-USART pins

The LIN-USART pins are shared with general purpose ports. Table 31-54 lists the pin functions, I/O formats and settings required to use the LIN-USART.

Table 31-54. LIN-USART pins

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
Pxx_i/SIN	Port I/O or Serial Data Input	Refer to 18. Port Pin Configuration which describes how to select available formats	Programmable pull-up resistor (refer to 18. Port Pin Configuration for how to activate the pull-up resistor)	Provided	Set corresponding general purpose port to 'input'
Pyy_j/SOT	Port I/O or Serial Data Output				
Pzz_k/SCK	Port I/O or serial clock input/output				When a clock is input, then set corresponding general purpose port to 'input'

31.5 LIN-USART interrupts

The following section describes LIN-USART interrupts.

LIN-USART uses reception, transmission and error interrupts. An interrupt request can be generated for either of the following causes:

- A reception interrupt request when: data is set in the Reception Data Register (USARTn_RDR); or the bus is idle and no reception error has occured.
- A transmission interrupt request when: transmission data is transferred from the Transmission Data Register (USARTn_TDR) to the transmission shift register; the last bit of the transmission data was shifted out; and no transmission error has occured.
- An error interrupt request when a parity/framing/overrun error occurs with the data received or a bus error occurs.

A DMA request (for single transfer or demand transfer) can be generated for reception and transmission interrupts if the corresponding DMA request enables are set. Interrupt request and DMA request are sent as separate output signals. When a reception error occurs, RX-DMA request is masked, and when a transmission error occurs, TX-DMA is masked.

LIN-USART interrupts

Table 31-55. Interrupt control bits and interrupt causes of LIN-USART

Reception/ transmission /test mode	Interrupt request flag bit	Flag register	Operation mode				Interrupt cause	Interrupt cause enable bit	How to clear the interrupt request
			0	1	2	3			
Reception	RDRF	USARTn_SSR	o	o	o	o	Receive data is written to USARTn_RDR or receive FIFO reaches the receive FIFO trigger level	USARTn_SSR:RIEor USART-n_EIER:RXFIE (when receive FIFO is enabled)	Received data is read
	LBD	USARTn_ESCR	x	x	x	o	LIN sync break detected	USARTn_ESCR: LBIE	Cleared by writing '1' to USARTn_ESCCR: LBDC
	RBI, TBI	USART-n_ECCR, USAR-Tn_ESIR	o	o	o	o	Neither trans-mission nor reception activ-ity	USARTn_ECCR: BIE	Write '1' to USART-n_ESICR: RBIC, write data to USARTn_TDR
Error	ORE	USARTn_SSR	o	o	o	o	Overrun error	USARTn_ESIR: OREIE	Cleared by writing '1' to USARTn_SCR:CRE (Clear Receive Errors bit)
	FRE	USARTn_SSR	o	o	1	o	Framing error	USARTn_ESIR: FREIE	
	PE	USARTn_SSR	o	x	1	x	Parity error in data bit in modes 0 and 2	USARTn_ESIR: PEIE	
	BUSERR	USARTn_ESR	o	o	o	o	When bus error occurs	USARTn_EIER: BUSERRIE	Cleared by writing '1' to USARTn_ESCLR: BUSERRC
Transmission2	TDRE	USARTn_SSR	o	o	o	o	USARTn_TDR is empty or transmit FIFO reaches the transmit FIFO trigger level	USARTn_SSR:TIE or USART-n_EIER:TXFIE (when transmit FIFO is enabled)	Write data to USART-n_TDR
	LBSOF	USARTn_ESR	o	o	o	o	Transmitted last bit of data in sync mode and stop bit in async mode	USARTn_EIER: LBSOIE	Cleared by writing '1' to USARTn_ESCLR: LBSOFC or by writing next byte of data to USARTn_TDR
	TXHRI	USARTn_ESR	x	x	x	o	Automatic transmission of LIN frame header com-pleted	USARTn_EIER: TXHDIE	Cleared by writing '1' to USARTn_ESCLR: TXHRIC

o: Used

x: Unused

1: Only available if USARTn_ECCR:SSM = '1'.

2: It is not recommended to use bus idle interrupt with USARTn_ESIR:AICD = '0', because in this case, as soon as reception activity starts, this interrupt is cleared.

Note: All flags can also be cleared by USARTn_SMR:UPCL.

Reception interrupt

If one of the following events occurs in reception mode, the corresponding flag bit of the Serial Status Register (USARTn_SSR) and the Extended Serial Interrupt Register (USARTn_ESIR) is set to '1':

- Data reception is complete, i.e. the received data was transferred from the serial input shift register to the Reception Data Register (USARTn_RDR) and data can be read, USARTn_SSR:RDRF, USARTn_ESIR:RDRF
  If USARTn_ESIR:AICD = '0' and the flag bit USARTn_SSR:RDRF is set to '1', and the reception interrupt is enabled (USARTn_SSR:RIE = '1') or USARTn_EIER:RXFIE = '1' when receive FIFO is enabled), a reception interrupt request is generated. If USARTn_ESIR:AICD = '1' and the flag bit USARTn_ESIR:RDRF is set to '1' and the reception interrupt is enabled (USARTn_SSR:RIE = '1'), or USARTn_EIER:RXFIE = '1' when receive FIFO is enabled), a reception interrupt request is generated. If the Reception Data Register (USARTn_RDR) is read, the USARTn_SSR:RDRF flag is automatically cleared to '0'. This bit is also cleared by writing '1' to USARTn_SMR:UPCL bit. USARTn_ESIR:RDRF flag is cleared by writing '1' to its corresponding clear bit USARTn_ESICR: RDRFC.
- When LIN synchronization break is detected
  This is relevant if LIN-USART operates in Mode 3. If the bus (serial input) goes '0' (dominant) for more than 10.5-bit times, the LIN-Break Detected (USARTn_ESCR:LBD) flag bit is set to '1'. In this case after 9-bit times the reception error flags are set to '1', therefore the USARTn_SCR:RXE flag has to set to '0', if only a LIN sync break detection is desired. The interrupt and the USARTn_ESCR:LBD flag are cleared after writing a '1' to the USARTn_ESCCR:LBDC flag.
- When there is no event on the bus (i.e when bus idle occurs)
  Reception bus idle cannot be used under the following conditions:
  - Synchronous Slave mode (USARTn_SCRn_MD = ‘10’; USARTn_SCRn_MD = ‘10’)
  - Synchronous Master mode with non-continuous clock (USART_SCRn_MD = ‘11’; USARTn_ECCR:MS = ‘0’; USARTn_ESCR:CCO = ‘0’)
  - During reception of LIN-sync byte in LIN-Mode (USARTn_SCRn_MD = ‘11’)
    If no reception is ongoing at the SIN pin, the bits USARTn_ECCR:RBI and USARTn_ESIR:RBI are set to '1'. USARTn_ECCR:RBI is cleared as soon as reception starts, while USARTn_ESIR:RBI must be cleared explicitly by writing '1' to USARTn_ESICR:RBIC bit.
    If no transmission is ongoing at the SOT pin, the bit USARTn_ECCR:TBI is set to '1'. When transmission is starting, USARTn_ECCR:TBI is cleared.
    If USARTn_ESIR:AICD = '0' and USARTn_ECCR:BIE = '1', the reception interrupt is generated as soon as both USARTn_ECCR:RBI and USARTn_ECCR:TBI are '1'. It is cleared as soon as either USARTn_ECCR:RBI or USARTn_ECCR:TBI is cleared.

Note:

Do not use the bus idle interrupt with USARTn_ESIR:AICD = '0', because USARTn_ECCR:RBI may be cleared before the interrupt handling software is able to clear the interrupt. If USARTn_ESIR:AICD = '1' and USARTn_ECCR:BIE = '1', the interrupt is generated as soon as both USARTn_ESIR:RBI and USARTn_ECCR:TBI are '1'. It is cleared as soon as either USARTn_ECCR:RBI or USARTn_ECCR:TBI is cleared. Use the bus idle interrupt with USARTn_ESIR:AICD = '1', because both USARTn_ESIR:RBI and USARTn_ECCR:TBI are acknowledged by the interrupt service routine.

Error interrupt

If one of the following errors occur, the corresponding flag bit of the Serial Status Register (USARTn_SSR), and the Extended Status Register (USARTn_ESR) is set to '1'.

- Reception Errors:
  - Overrun error, i.e. USARTn_SSR:RDRF = '1' and USARTn_RDR was not read by the CPU and received next serial data: USARTn_SSR:ORE
  - Framing error, i.e. a stop bit was expected, but a '0'-bit was received: USARTn_SSR:FRE

- Parity error, i.e. a wrong parity bit was detected: USARTn_SSR:PE
  Note: Reception error is not asserted if any reception error flag is set.

- Transmission Errors:
  - Bus error during detection of physical bus error USARTn_ESR:BUSERR in LIN mode

If one of the flag bits and corresponding interrupt enable bits as mentioned in Table 31-55 is set to '1', an error interrupt is generated

The error interrupt and these flag bits are cleared by writing '1' to their corresponding clear bits. For details refer to Table 31-55.

Transmission interrupt

A transmission interrupt is generated if one of the following events occur:

- If transmission data is transferred from the Transmission Data Register (USARTn_TDR) to the Transmission Shift Register, the Transmission Data Register Empty Flag bit (USARTn_SSR:TDRE) is set to '1'. In this case, an interrupt request is generated if the Transmission Interrupt Enable (USARTn_SSR:TIE or USARTn_EIER:TXFIE when transmit FIFO is enabled) bit was set to '1'.

Notes:

1. The initial value of USARTn_SSR:TDRE (after hard or soft reset) is '1'. So an interrupt is generated immediately if the USARTn_SSR:TIE is set to '1' or USARTn_EIER:TXFIE is set to '1' when the transmit FIFO is enabled.
2. The USARTn_SSR:TDRE flag is initialized (set to ‘1’) by the following conditions:
   - Writing data to the Transmission Data Register (USARTn_TDR)
   - Changing the operation mode of the USART
   - Writing ‘1’ to USARTn_SCR:UPCL

- If the last bit of data in sync mode or a stop bit in async mode is shifted out (USARTn_ESR:LBSOF = '1'), then an interrupt request is generated if the interrupt enable USARTn_EIER:LBSOIE is set to '1'. The interrupt and the flag bits are cleared by writing '1' to USARTn_ESCLR:LBSOFC bit

Note: Do not use USARTn_SSR:TIE and USARTn_EIER:LBSOIE together at the same time. This would cause two interrupt requests for the same transmitted byte.

- When header transmission is completed, USARTn_ESR:TXHRI = ‘1’ and the corresponding interrupt enable bit USARTn_EIER:TXHDIE is set to '1' and a transmission interrupt is generated.

31.5.1 Reception interrupt generation and flag set timing

The reception interrupt is caused by the completion of reception (USARTn_SSR:RDRF).

Reception interrupt generation and flag set timing

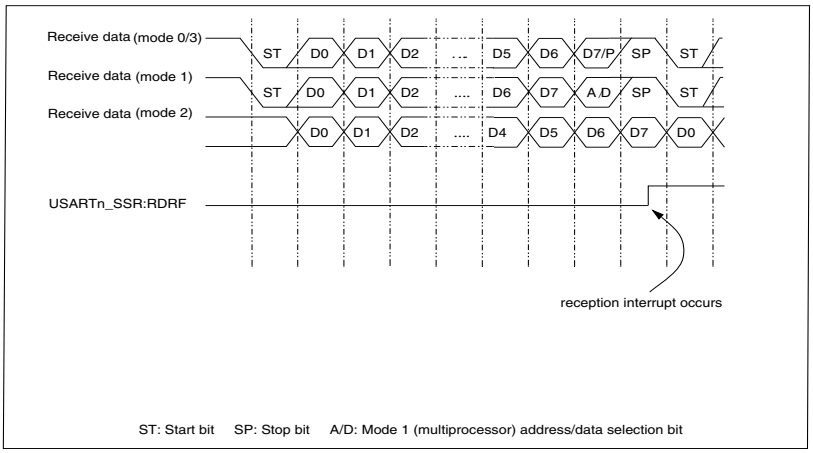
Generally a reception interrupt is generated, if the received data is complete (when USARTn_ESIR:AICD = '0', USARTn_SSR:RDRF = '1', when USARTn_ESIR:AICD = '1', USARTn_ESIR:RDRF = '1') and the Reception Interrupt Enable bit (without FIFO, then USARTn_SSR:RIE; and with FIFO, then USARTn_EIER:RXFIE) is set to '1'. This interrupt is generated, if the first stop bit is detected in mode 0, 1, 2 (if USARTn_ECCR:SSM = '1'), 3, or the last data bit was read in mode 2 (if USARTn_ECCR:SSM = '0').

Notes:

If a reception error has occurred,

- The Reception Data Register (USARTn_RDR) contains invalid data in each mode
- The USARTn_SSR:RDRF and USARTn_ESIR:RDRF flags are not set (also not cleared if already set) and so the receive interrupt is not set

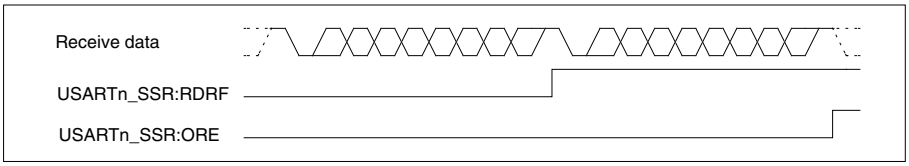
Figure 31-52. Reception operation and flag set timing (when reception FIFO is disabled)



The example in Figure 31-52 does not show all possible reception options for mode 0 and 3. Here it is '7p1' and '8N1' (p = 'E' [even] or 'O' [odd]). When the reception FIFO is enabled (USARTn_RFCR:RXFE set to '1'), the USARTn_SSR:RDRF flag will be asserted when the FIFO reaches the trigger level programmed in the Reception FIFO Control Register (USARTn_RFCR:RXFLC). A reception interrupt is generated in this case when USARTn_EIER:RXFIE is set to '1'.

Error interrupt generation and flag set timing

Figure 31-53. USARTn_SSR:ORE set timing (when reception FIFO is disabled)



Note: When reception FIFO is enabled (USARTn_RFCR:RXFE is set to '1'), USARTn_SSR:ORE flag will be set when reception FIFO overflows (i.e) when the reception FIFO is full (16 bytes already received) and another data byte is received.

31.5.2 Transmission interrupt generation and flag set timing

A transmission interrupt is generated when the transmission data is transferred from Transmission Data Register (USARTn_TDR) to transmission shift register or when last bit was shifted out.

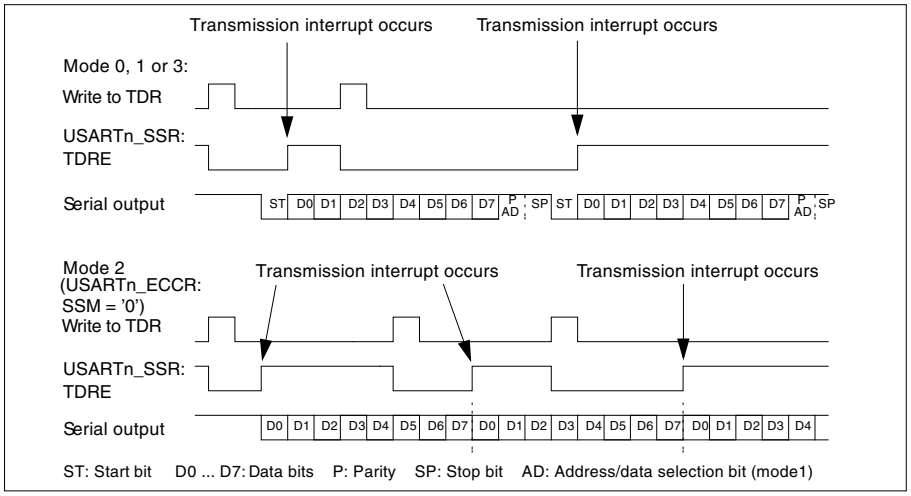
Transmission interrupt generation and flag set timing using USARTn_SSR:TIE

A transmission interrupt is generated, when the next data to be sent is ready to be written to the Transmission Data Register (USARTn_TDR), i.e. the USARTn_TDR is empty, and the transmission interrupt is enabled by setting the Transmission Interrupt Enable (without FIFO, then USARTn_SSR:TIE; and with FIFO USARTn_EIER:TXFIE) bit to '1'.

The Transmission Data Register Empty (USARTn_SSR:TDRE) flag bit indicates an empty USARTn_TDR. Writing data to USARTn_TDR clears the USARTn_SSR:TDRE bit.

Figure 31-54 demonstrates the transmission operation and flag set timing for the four modes of LIN-USART.

Figure 31-54. Transmission operation and flag set timing (when transmission FIFO is disabled)



Notes:

1. The example in Figure 31-54 does not show all possible transmission options for mode 0. Here it is, '8p1' (p = 'E' [even] or 'O' [odd]). Parity is not provided in mode 3 or 2, if USARTn_ECCR:SSM = '0'.
2. When transmission FIFO is enabled, the USARTn_SSR:TDRE flag will be asserted when the FIFO reaches the trigger level programmed in the Transmission FIFO Control Register (USARTn_TFCR:TXFLC). Transmission interrupt is generated in this case when USARTn_EIER:TXFIE is set to '1'.

Transmission interrupt request generation timing using USARTn_EIER:LBSOIE

If the USARTn_ESR:LBSOF flag is set to '1', the interrupt is enabled (USARTn_EIER:LBSOIE = '1') and a transmission interrupt request is asserted when the last bit is shifted out.

Figure 31-55 below shows the timing for asynchronous mode with one stop bit. USARTn_ESR:LBSOF is set at the end of the bit time of the stop bit.

Figure 31-55. Transmission operation and flag set timing for USARTn_ESR:LBSOF in asynchronous mode

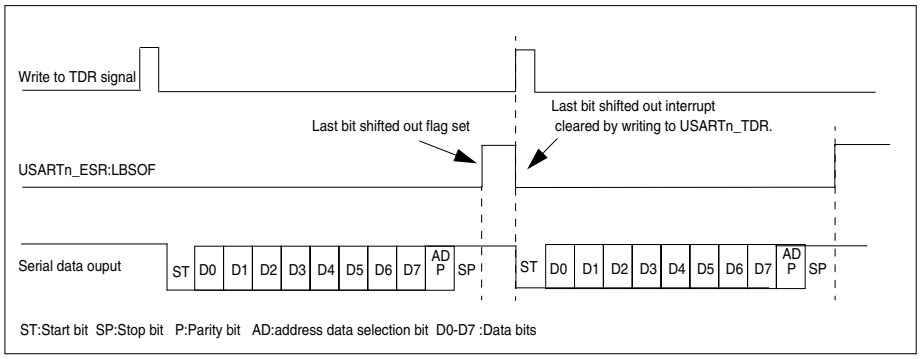


Figure 31-56 shows the flag set timing in synchronous master mode and synchronous slave mode with continuos clock (USARTn_ESCR:CCO = '1'). USARTn_ESR:LBSOF is set at the end of the bit time of the last transferred bit.

Figure 31-56. USARTn_ESR:LBSOF set timing in synchronous master mode and synchronous slave mode with continuos clock

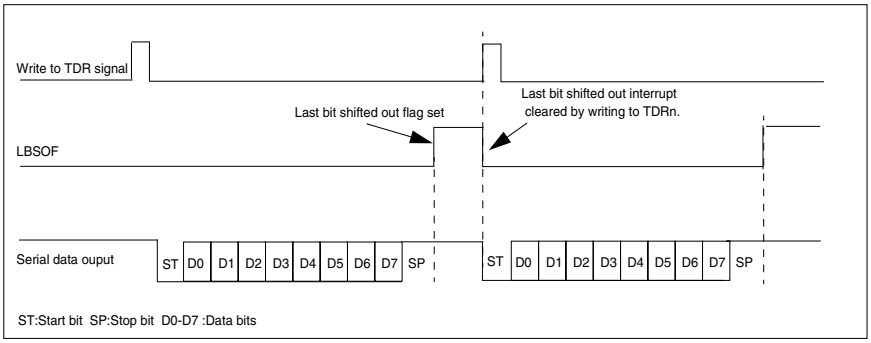
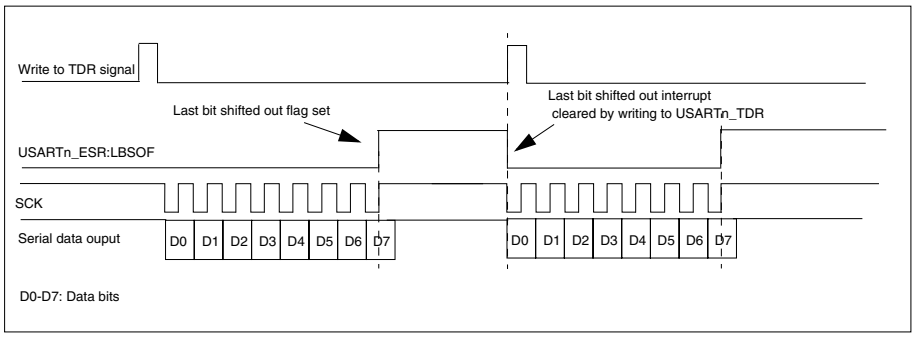


Figure 31-57 shows the flag set timing in synchronous slave mode without continuos clock (USARTn_ESCR:CCO = '0'). USARTn_ESR:LBSOF is set after half of the bit time of the last transferred bit.

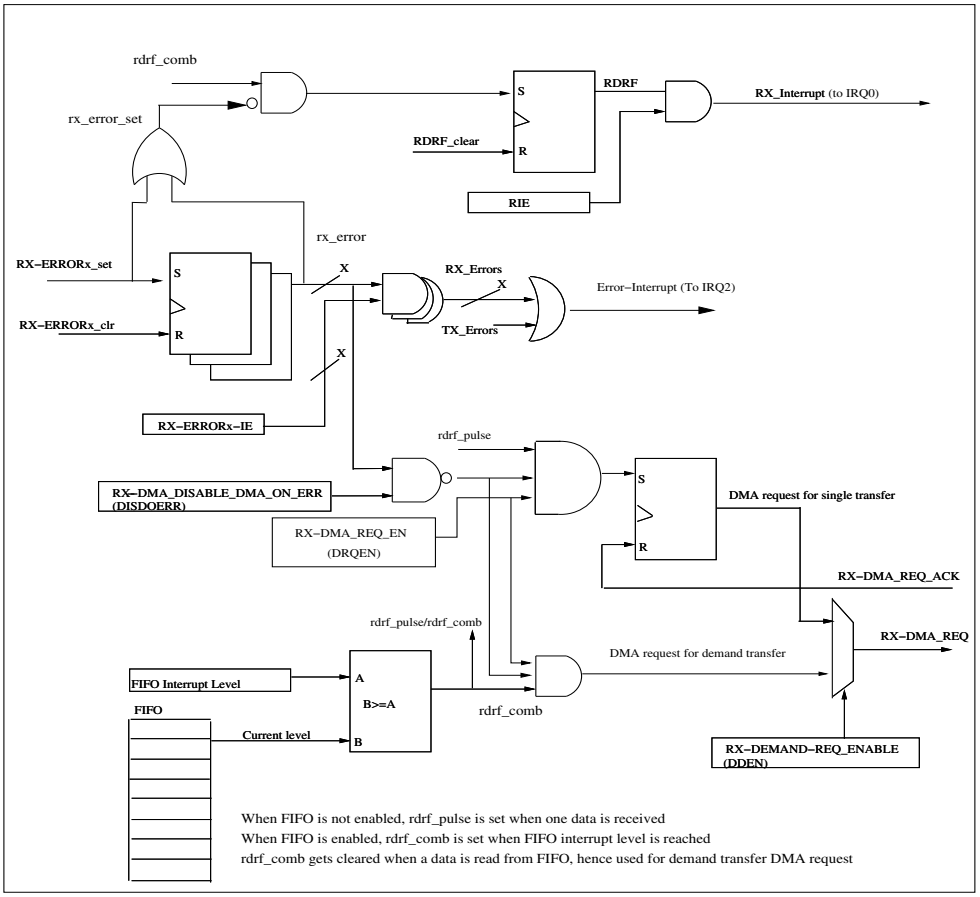
Figure 31-57. USARTn_ESR:LBSOF set timing in synchronous slave mode with non-continuos clock



31.6 LIN-USART DMA functions

The LIN-USART can operate with DMA for reception or transmission. DMA requests are generated as separate output signals rather than the interrupt request signals. With reference to Figure 31-58, the DMA requests (DMA_REQ_RX and DMA_REQ_TX) have their corresponding DMA request enable bits (USARTn_SRXDR:RXDRQEN and USARTn_STXDR:TX DRQEN respectively).

Figure 31-58. DMA requests are generated as seperate output signals



If a reception error occurs and the ‘Disable DMA on error’ bit is set (USARTn_SRXDR:RXDISDOERR = ‘1’), the DMA _REQ_RX request is masked as long as the error flag is set.

If a transmission error occurs and the ‘Disable DMA on error’ bit is set (USARTn_STXDR:TXDISDOERR = ‘1’), the DMA _REQ_TX request is masked as long as the error flag is set.

Any DMA request can be of two types: Single transfer with acknowledge or demand transfer (chosen by USARTn_S-RXDR:RXDDEN for DMA_REQ_RX and USARTn_STXDR:TXDDEN for DMA_REQ_TX).

- Single transfer (must be used if FIFO is disabled)
  - No DMA is requested if the error occurs in the byte received
  - If the error is later cleared, the DMA transfer is not requested even if the RDR is full or the RX-FIFO level is above the RX-FIFO trigger level. The RDR must be cleared before the next data is received in order to get the new DMA request
  - If USARTn_SCR:SBL= ‘1’ (2 STOP bits), then the DMA request is not masked if the Framing error is in the 2nd STOP bit

- The DMA request for single transfer is cleared by the DMA request acknowledge (DMA_REQ_ACK_RX for DMA_REQ_RX and DMA_REQ_ACK_TX for DMA_REQ_TX) and clearing of DMA request has high priority over setting the request.
- When any reeption error (USARTn_SSR:PE, USARTn_SSR:ORE, USARTn_SSR:FRE) occurs, the DMA_REQ_RX is masked if USARTn_SRXDR:RXDISDOERR is set. Similarly a transmission error (USARTn_ESR:BUSERR) masks DMA_REQ_TX if USARTn_STXDR:TXDISDOERR is set.

- Demand transfer (must be used if FIFO is enabled)
  - The DMA request during transmission is set as long as the TX-FIFO level is below the configured TX-FIFO interrupt level.
  - The DMA request during reception is set as long as the RX-FIFO level is above the configured RX-FIFO interrupt level.
  - When a reception/transmission error occurs, the demand transfer is also masked similar to the single transfer request

For more details on 'DMA' refer to 41. DMA Controller.

Notes:

1. If DMA is used (USARTn_SRXDR:RXDRQEN or USARTn_STXDR:TXDRQEN is '1') SW should not enable the corresponding interrupt request (USARTn_SSR:RIE or USARTn_SSR:TIE) else ISR will be entered in addition to DMA transfer.
2. If disable DMA on error (USARTn_SRXDR:RXDISDOERR for receive errors and USARTn_STXDR:TXDISDOERR for transmit errors) is enabled, error interrupt should be enabled to handle the error conditions.
3. If disable DMA on error (USARTn_SRXDR:RXDISDOERR for receive errors and USARTn_STXDR:TXDISDOERR for transmit errors) is not enabled, wrong (errored) data might be transferred by DMA.

LIN-USART DMA masking

Table 31-56. DMA requests and mask

DMA request	DMA request trigger condition	DMA Request Enable	Errors capable of masking the DMA request	DMA Mask Enable
DMA_REQ_RX	If FIFO is disabled, the USARTn_RDR is loaded with the received data. If FIFO is enabled, the number of bytes in the receive FIFO is greater than or equal to the received FIFO interrupt trigger level set in USARTn_RFCR:RXFLC[4:0]	USARTn_SRXDR:RXDRQEN	Parity Error in Data (USARTn_SSR:PE) Framing Error (USARTn_SSR:FRE) Overrun Error (USARTn_SSR:ORE)	USARTn_SRXDR:RXDISDOERR
DMA_REQ_TX	If FIFO is disabled, USARTn_TDR is empty. If FIFO is enabled, the number of bytes in transmit FIFO is less than or equal to the Transmit FIFO interrupt trigger level set in USARTn_TFCR:TXFLC[4:0]	USARTn_STXDR:TXDRQEN	Bus Error (USARTn_ESR: BUSERR)	USARTn_STXDR:TXDISDOERR

31.7 LIN-USART baud rates

One of the following can be selected for the LIN-USART serial clock source:

- Dedicated baud rate generator (reload counter)
- External clock as it is (clock input to the SCK pin)
- External clock connected to the baud rate generator (reload counter)

LIN-USART baud rate selection

The baud rate selection circuit is designed as shown below. One of the following three types of baud rates can be selected:

- Baud rates determined using the dedicated baud rate generator (reload counter)

  LIN-USART has two independent internal reload counters for transmission and reception serial clock. The baud rate can be selected via the 19-bit reload value determined by the Baud Rate Generator Reload Registers (USARTn_BGRL)

  The reload counter divides the peripheral clock, CLK, by the value set in the Baud Rate Generator Registers.
- Baud rates determined using external clock (one-to-one mode)

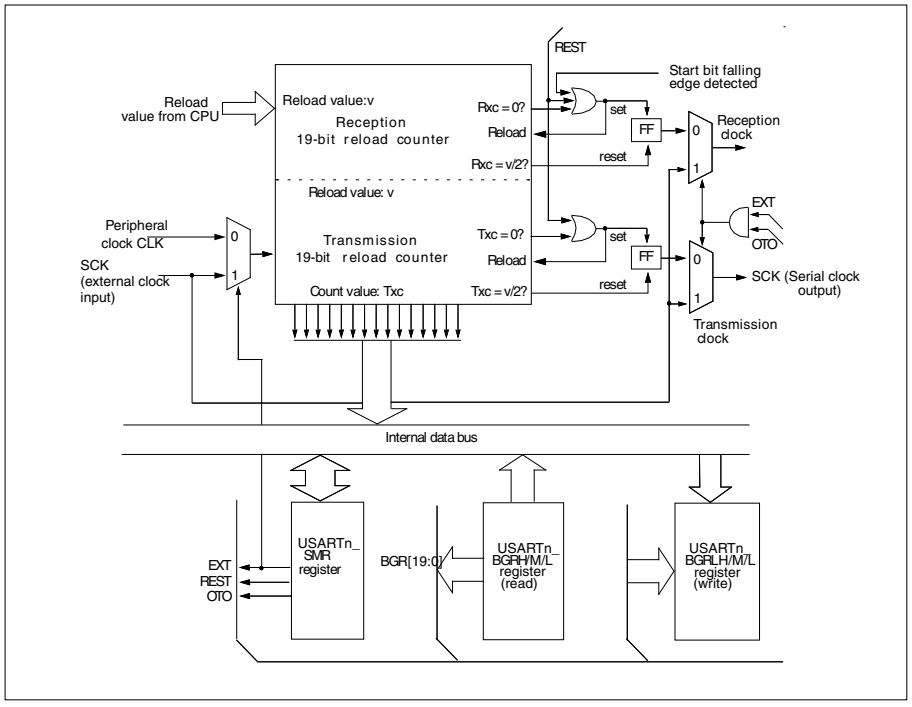
  The clock input from LIN-USART clock pulse input pins (SCK) is used as it is (synchronous). Any baud rate less than the peripheral clock CLK divided by 5 and is divisible can be set externally.
- Baud rates determined using the dedicated baud rate generator with external clock

  An external clock source can also be connected internally to the reload counter. In this mode it is used instead of the internal peripheral clock, CLK. This was designed to use quartz oscillators with special frequencies and having the possibility to divide them.
- Baud rates for asynchronous mode

  For asynchronous communication, the peripheral clock CLK value must be equal to or greater than 4 times the baud rate.
- Baud rates for synchronous mode

  For synchronous communication, he peripheral clock CLK value must be equal to or greater than 5 times the baud rate.

Figure 31-59. Baud rate selection circuit (reload counter)



31.7.1 Setting the baud rate

This section describes how the baud rates are set and the resulting serial clock frequency is calculated

Calculating the baud rate

Both the 19-bit reload counters are programmed by the Baud Rate Generator Reload Registers (USARTn_BGRL). The following calculation formula should be used to set the desired baud rate:

Reload value: v = [Φ / b] - 1,

where,

Φ is the peripheral clock CLK

b is the baud rate

[ ] is the gaussian brackets (mathematical rounding function)

Example of calculation

If the peripheral clock CLK is 16 MHz and the desired baud rate is 19,200 baud, then the reload value v is:

$v = [16*10^6 / 19200] - 1 = 832$

The exact baud rate can then be recalculated: $b_{exact} = \Phi / (v + 1)$, here it is:

$16*10^6 /833 = 19207.6831$

Note: Setting the reload value to '0' stops the reload counter. For this reason the minimum division ratio is 2.

Suggested division ratios for different peripheral clock CLK frequencies and baud rates

The following settings are suggested for different peripheral clock CLK frequencies and baud rates:

Table 31-57. Suggested baud rates and reload values at different peripheral clock CLK frequencies

Baud rate	16 MHz		24 MHz		32 MHz		48 MHz*		56 MHz*		72 MHz*	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
12M	-	-	-	-	-	-	-	-	-	-	5	0
8M	-	-	-	-	-	-	5	0	6	0	8	0
4M	-	-	5	0	7	0	11	0	13	0	17	0
2M	7	0	11	0	15	0	23	0	27	0	35	0
1M	15	0	23	0	31	0	47	0	55	0	71	0
500000	31	0	47	0	63	0	95	0	111	0	143	0
460800	-	-	51	-0.16	68	-0.6	103	-0.16	120	-0.43	155	-0.16
250000	63	0	95	0	127	0	191	0	223	0	287	0
230400	-	-	103	-0.16	137	-0.6	207	-0.16	242	-0.02	312	-0.16
153600	103	-0.16	155	-0.16	207	-0.16	311	-0.16	363	-0.16	468	0.05
125000	127	0	191	0	255	0	383	0	447	0	575	0
115200	138	0.08	207	-0.16	276	-0.2	415	-0.16	485	-0.02	624	0
76800	207	-0.16	311	-0.16	416	0.079	624	0	728	-0.02	937	0.05
57600	277	0.08	416	0.08	554	-0.1	832	-0.04	971	-0.02	1249	0
38400	416	0.08	624	0	832	-0.04	1249	0	1457	-0.02	1874	0
28800	554	-0.01	832	-0.03	1110	-0.01	1665	-0.04	1943	-0.02	2499	0

Table 31-57. Suggested baud rates and reload values at different peripheral clock CLK frequencies

Baud rate	16 MHz		24 MHz		32 MHz		48 MHz*		56 MHz*		72 MHz*	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
19200	832	-0.03	1249	0	1665	-0.04	2499	0	2915	-0.02	3749	0
10417	1535	<0.01	2303	<0.01	3070	-0.02	4606	-0.01	5374	-0.015	6911	0.0032
9600	1666	0.02	2499	0	3332	-0.01	4999	0	5832	-0.005	7499	0
7200	2221	<0.01	3332	<0.01	4443	-0.01	6665	-0.01	7776	-0.01	9999	0
4800	3332	<0.01	4999	0	6665	-0.01	9999	0	11665	-0.005	14999	0
2400	6666	<0.01	9999	0	13332	-0.002	19999	0	23332	-0.001	29999	0
1200	13332	<0.01	19999	0	26665	-0.002	39999	0	46665	-0.001	59999	0
600	26666	<0.01	39999	0	53332	-0.0006	79999	0	93332	-0.0004	119999	0
300	53332	-0.0006	79999	0	106666	0.0003	159999	0	186666	0.0002	239999	0

Notes:

1. Deviations (“dev.” in the table above) are given in %.
2. The maximum synchronous baud rate: peripheral clock CLK frequency divided by 5.
3. *: These frequencies are applicable only for USART6 to USART11.

Using external clock

If the EXT bit of the USARTn_SMR register is set, an external clock is selected, which has to be connected to the SCK pin. The external clock is used in the same way as the peripheral clock CLK to the baud rate reload counter.

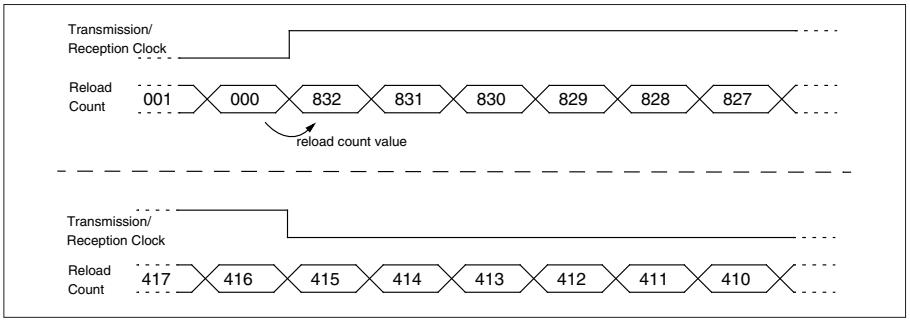
If one-to-one external clock input mode (USARTn_SMR:OTO) is selected, the SCK signal is directly connected to the LIN-USART serial clock inputs. This is needed for the LIN-USART synchronous mode 2 operating as slave device.

Note: In any case the resulting clock signal is synchronized to the peripheral clock CLK in the LIN-USART module. This means that indivisible clock rates will result in phase unstable signals.

Counting example

Assume the reload value is 832. Figure 31-60 demonstrates the behavior of both the reload counters.

Figure 31-60. Counting example of the reload counters



Note: The falling edge of the serial clock signal always occurs after | (v + 1)/2 |.

31.7.2 Restarting the reload counter

The reload counters can be restarted of the following reasons:

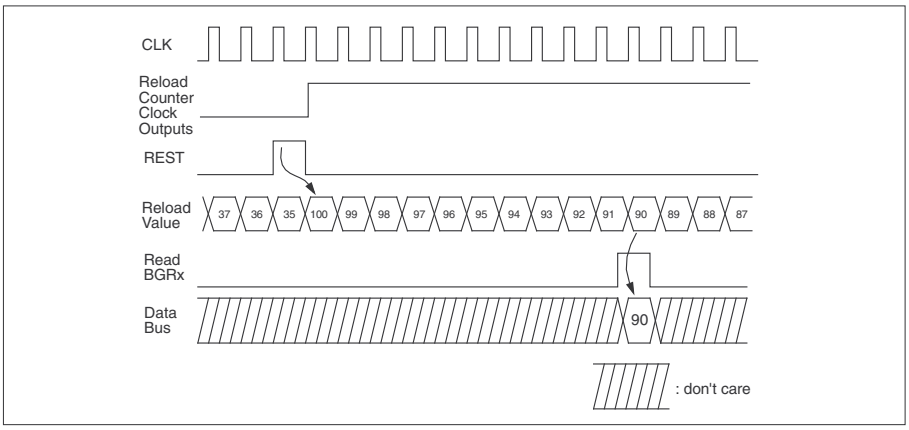
- Transmission and reception reload counter
  - Global MCU reset
  - LIN-USART programmable clear (USARTn_SMR:UPCL bit)
  - User programmable restart (USARTn_SMR:REST bit)
- Reception reload counter
  - Start bit falling edge detection in asynchronous mode

Programmable restart

If the USARTn_SMR:REST bit of the Serial Mode Register (USARTn_SMR) is set by the user, both reload counters are restarted at the next peripheral clock cycle (CLK). This feature is intended to use the transmission reload counter as a small timer.

The following figure illustrates a possible usage of this feature (assume that the reload value is 100.)

Figure 31-61. Reload counter restart example



In this example the number of peripheral clock CLK cycles (cyc) after USARTn_SMR:REST is then:

cyc = v - c + 1 = 100 - 90 + 1 = 11

where,

v is the reload value

c is the read counter value

Note: If LIN-USART is reset by setting USARTn_SMR:UPCL, the reload counters will restart too.

- Automatic restart

In asynchronous LIN-USART mode, if a falling edge of a start bit is detected, the reception reload counter is restarted. This is intended to synchronize the serial input shifter to the incoming serial data stream. Additionally a restart is performed, if a LIN break falling edge is detected when USARTn_EFER:LBEDGE is selected.

Clearing reload counters

The Baud Rate Reload Register (USARTn_BGR) and the baud rate reload counters are cleared to '0' by the MCU global reset and the counters stops. The reload counters are cleared to '0' by writing '1' to the USARTn_SMR:UPCL bit. However, the value stored in the reload register is kept unchanged and the counters start from reload value immediately.

31.8 Operation of the LIN-USART

LIN-USART operates in operation mode 0 for normal bidirectional serial communication, in mode 2 and 3 for bidirectional communication as master or slave, and in mode 1 as master or slave in multiprocessor communication.

Operation of LIN-USART

- Operation modes

There are four LIN-USART operation modes: modes 0 to 3. As listed in Table 31-58, an operation mode can be selected according to the communication method.

Table 31-58. LIN-USART operation mode

Operation mode		Data length		Synchronization of mode	Length of stop bit	Data bit direction *1
		Parity disabled	Parity enabled			
0	Normal mode	7 or 8		Asynchronous	1 or 2	L/M
1	Multipro-cessor	7 or 8 + 1*2	-	Asynchronous	1 or 2	L/M
2	Normal mode	8 or 9 (if USARTn:ECCR: SSM = '1')		Synchronous	0, 1 or 2	L/M
3	LIN mode	8	-	Asynchronous	1	L

*1: means the data bit transfer format: LSB or MSB first

*2: '+1' means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

Note:

Mode 1 operation is supported both for master or slave operation of LIN-USART in a master-slave connection system. In mode 3 the LIN-USART function is locked to 8N1-format, LSB first.

If the mode is changed, LIN-USART cuts off all possible transmission or reception and awaits the new action. It is strongly recommended to reset the USART (write '1' to SMRn:UPCL) after changing operation mode.

Inter-CPU connection method

External clock one-to-one connection (normal mode) and master-slave connection (multiprocessor mode) can be selected. For either connection method, the data length, whether to enable parity, and the synchronization method must be common to all CPUs. Select an operation mode as follows:

In the one-to-one connection method, operation mode 0 or 2 must be used in the two CPUs. Select operation mode 0 for asynchronous transfer mode and operation mode 2 for synchronous transfer mode.

Notes:

1. One CPU has to be configured as master and the other has to be configured as slave in synchronous mode 2.
2. Select operation mode 1 for the master-slave connection method and use it either for the master or slave system.

Synchronization methods

In asynchronous operation LIN-USART reception clock is automatically synchronized to the falling edge of a received start bit (reception baud rate counter is restarted).

In synchronous mode the synchronization is performed either by the clock signal of the master device or by LIN-USART itself if operating as master.

Signal mode

USART can treat data in normal or inverted format,

Operation enable bit

LIN-USART controls both transmission and reception using the operation enable bit for transmission (USARTn_SCR:TXE) and reception (USARTn_SCR:RXE).

- If reception should be disabled, wait until the byte is completely received and read the received data from Reception Data Register (USARTn_RDR). Then disable reception by clearing USARTn_SCR:RXE.
- If transmission should be disabled, wait until byte is completely transmitted and ensure that Transmission Data Register is empty (USARTn_TRE). Then disable transmission by clearing USARTn_SCR:TXE.

31.8.1 Operation in asynchronous mode (operation modes 0 and 1)

When LIN-USART is used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode), the asynchronous transfer mode is selected.

31.8.1.1 Operation in asynchronous mode

Transfer data format

Generally each data transfer in the asynchronous mode operation begins with the start bit (low- level on bus) and ends with at least one stop bit (high-level). The direction of the bit stream (LSB first or MSB first) is determined by the USARTn_SSR:BDS. The parity bit (if enabled) is always placed between the last data bit and the (first) stop bit.

In operation mode 0 the length of the data frame can be 7 or 8 bits, with or without parity, and 1 or 2 stop bits.

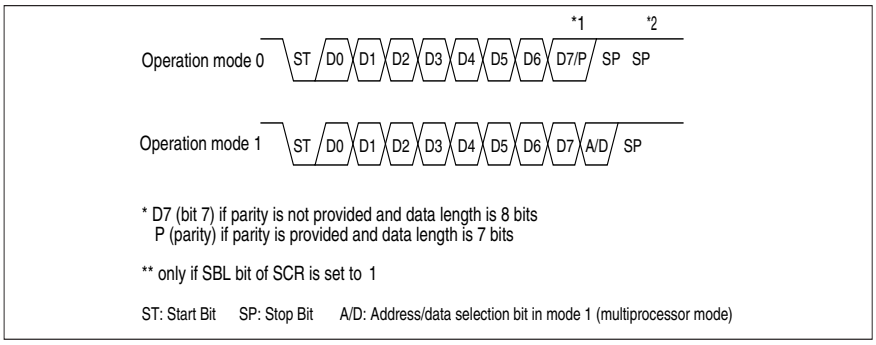
In operation mode 1 the length of the data frame can be 7 or 8 bits with a following address-/data- selection bit instead of a parity bit. 1 or 2 stop bits can be selected.

The calculation formula for the bit length of a transfer frame is:

Length = 1 + d + p + s

(d = number of data bits [7 or 8], p = parity [0 or 1], s = number of stop bits [1 or 2]

Figure 31-62. Transfer data format (operation modes 0 and 1))



Note:

If USARTn_SSR:BDS bit is set to '1' (MSB first), the bit stream processes as: D7, D6,..., D1, D0, (P).

During reception both stop bits are detected, if selected. But the Reception Data Register Full (USARTn_SSR:RDRF) flag will go '1' at the first stop bit. The bus idle flag (USARTn_ECCR:RBI) goes '1' after the second stop bit if no further start bit is detected. (The second stop bit belongs to 'bus activity', although it is just mark level.)

Transmission operation

If the Transmission Data Register Empty flag (USARTn_SSR:TDRE) is '1', transmission data is allowed to be written to the Transmission Data Register (USARTn_TDR). When data is written, the USARTn_SSR:TDRE flag goes '0'. If the transmission operation is enabled by the USARTn_SCR:TXE bit ('1'), the data is written next to the transmission shift register and the transmission starts at the next clock cycle of the serial clock, beginning with the start bit. Thereby the USARTn_SSR:TDRE flag goes '1', so that new data can be written to the USARTn_TDR.

If transmission interrupt is enabled (USARTn_SSR:TIE = '1' or (USARTn_EIER:TXFIE = '1', if TX FIFO is enabled)), the interrupt is generated by the USARTn_SSR:TDRE flag. Note that, the initial value of the USARTn_SSR:TDRE flag is '1', so that in this case if USARTn_SSR:TIE or USARTn_EIER:TXFIE (if TX FIFO is enabled) is set to '1' an interrupt occurs immediately.

When the character length is set to 7 bits (USARTn_SCR:CL = '0'), the unused bit of the USARTn_TDR is always the MSB, independently from the bit direction setting in the USARTn_SSR:BDS bit (LSB first or MSB first).

Reception operation

Reception operation is performed when it is enabled by the Reception Enable flag (USARTn_SCR:RXE). If a start bit is detected, a data frame is received according to the format specified by USARTn_SCR. In case of errors, the corresponding error flags are set (USARTn_SSR:PE, USARTn_SSR:ORE, USARTn_SSR:FRE). After the reception of the data frame, the data is transferred from the reception shift register to the Reception Data Register (USARTn_RDR) (and respectively to FIFO if RX-FIFO is enabled) and the Reception Data Register Full flags (USARTn_SSR:RDRF or USARTn_ESIR:RDRF) are set. If RX-FIFO is enaled the Reception Data full flags are set when the configured FIFO level is reached.

If receive interrupt is enabled (USARTn_SSR:RIE = '1' or (USARTn_EIER:RXFIE = '1', if RX FIFO is enabled)) and USARTn_ESIR:AICD = '0', the interrupt is generated by USARTn_SSR:RDRF. If receive interrupt is enabled (USARTn_SSR:RIE = '1' or (USARTn_EIER:RXFIE = '1', if RX FIFO is enabled)) and USARTn_ESIR:AICD = '1', the interrupt is generated by USARTn_ESIR:RDRF. The data then has to be read by the CPU. By doing so, the USARTn_SSR:RDRF flag is cleared. When USARTn_ESIR:AICD = '0', this also clears the interrupt.

When USARTn_ESIR:AICD = '1', writing '1' to USARTn_ESICR:RDRFC clears the interrupt

When the character length is set to 7 bits (USARTn_SCR:CL = '0'), the unused bit of the USARTn_RDR is always the MSB, independently from the bit direction setting in the USARTn_SSR:BDS bit (LSB first or MSB first).

Note: Only when the USARTn_SSR:RDRF flag is set and no errors have occurred does the Reception Data Register USARTn_RDR contain valid data.

Used clock

Use the internal clock or external clock. Select the baud rate generator (USARTn_SMR:EXT = '0' or '1', USARTn_SMR:OTO = '0') for desired baud rate.

Stop bit, error detection, and parity

Number of stop bit, 1 or 2 can be specified by the USARTn_SCR:SBL bit. When receiving and 2-bit is set to the stop bit, the second stop bit is checked in addition to the first stop bit. The RBI (bus idle) flag is set after the second stop bit. However, the USARTn_SSR:RDRF flag is set when the first stop bit is received. In mode 0, parity error, overrun error and framing error are checked. In mode 1, parity check is not supported and overrun error and framing error are checked. The USARTn_SCR:PEN bit of the USARTn_SCR register enables/disables the parity bit and the P bit specifies even or odd parity in mode 0.

31.8.2 Operation in synchronous mode (operation mode 2)

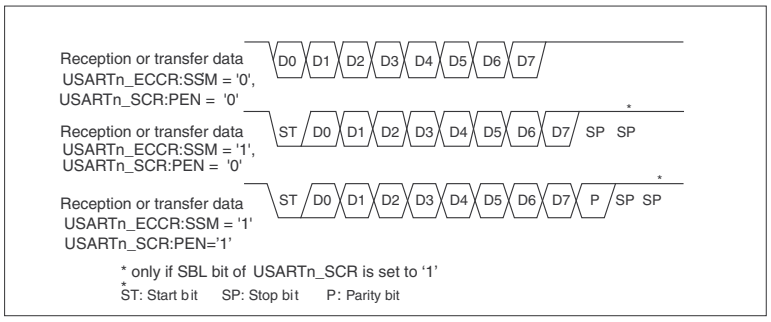
The clock synchronous transfer method is used for LIN-USART operation mode 2 (normal mode).

31.8.2.1 Operation in synchronous mode (operation mode 2)

Transfer data format

In the synchronous mode, 8-bit data is transferred without start or stop bits if the USARTn_ECCR:SSM bit of the Extended Communication Control Register (USARTn_ECCR) is '0'. The following figure illustrates the data format during a transmission in the synchronous operation mode.

Figure 31-63. Transfer data format (operation mode 2)



Clock inversion and start/stop bits in mode 2

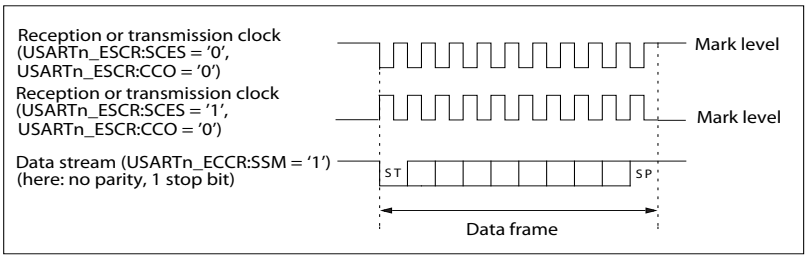
If the USARTn_ESCR:SCES bit is set the serial clock is inverted. Therefore in slave mode LIN- USART samples the data bits at the falling edge of the received serial clock.

Note:

In master mode if USARTn_ESCR:SCES is set, the clock signal's mark level is '0'.

If the USARTn_ECCR:SSM bit is set, the data format gets start and stop bits like in asynchronous mode.

Figure 31-64. Transfer data format with clock inversion

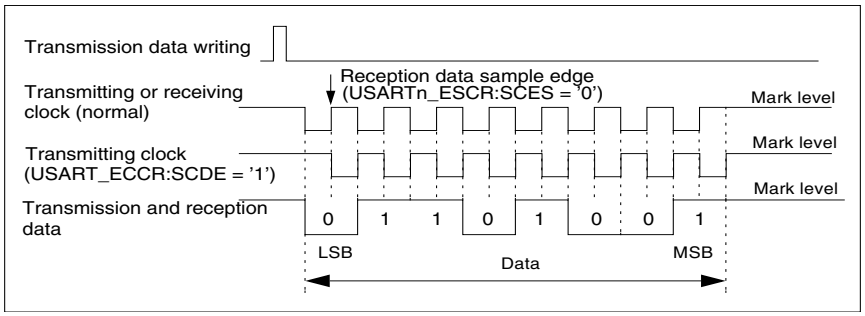


Clock supply

In operation mode 2, the number of clock cycles for the clock signal must be the same as the number of bits for the data including enabled start bit, parity bit and stop bits. If the USARTn_ECCR:MS bit is '0' (master mode), the consistent clock cycles are generated automatically. If the USARTn_ECCR:MS bit of is '1' (slave mode), ensure that correct clock cycles are generated by the master communication device. While there is no communication, the clock signal must be kept at '1' as the mark level.

If the USARTn_ECCR:SCDE bit is '1', the clock output signal is delayed by the half of the serial clock cycle as shown in Figure 31-65. The operation is prepared for communication devices which use the falling edge of the serial clock signal for data sampling.

Figure 31-65. Delayed transmitting clock signal (USARTn_ECCR:SCDE = '1')



If the USARTn_ESCR:SCES bit is '1', the serial clock signal is inverted. Receiving data is sampled at the falling edge of the serial clock.

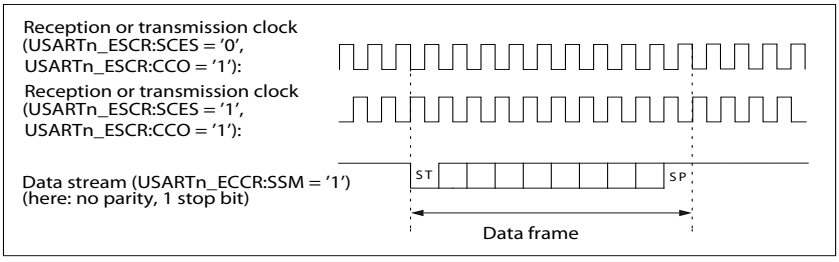
If the USARTn_ECCR:MS bit is '0' (master mode), the output clock signal is also inverted. While there is no communication, the clock signal must be kept at '0' as the mark level.

If the USARTn_ESCR:CCO bit is '1', the serial clock is signaled even while there is no data communication. Therefore, it is recommended to specify the start/stop bits as shown in Figure 31-66.

Table 31-59. Serial Data Input sampling depending on USARTn_ECCR:SCDE and USARTn_ESCR:SCES

Sr. no	USARTn_ECCR: SCDE	USARTn_ESCR: SCES	Serial data sampling edge	Serial data transmitting edge
1	'0'	'0'	Rising edge	Falling edge
2	'0'	'1'	Falling edge	Rising edge
3	'1'	'0'	Falling edge	Rising edge
4	'1'	'1'	Rising edge	Falling edge

Figure 31-66. Continuous clock output in mode 2



Note: If the USART set to Slave (MS=1) and CCO=1 and TDR is empty, one byte 0xFF is transferred as soon as the serial clock is supplied.

Error detection

If no start/stop bits are selected (USARTn_ECCR:SSM = '0') only overrun errors are detected. If parity error is enabled (USARTn_SCR:PEN='1'), then a parity error can also be detected.

31.8.3 Operation of LIN-USART (operation mode 3)

LIN-USART can be used as the LIN-master. A special mode is provided for this LIN function. Setting the LIN-USART to mode 3 configures the data format to 8N1-LSB-first format.

31.8.3.1 Features of LIN-USART in LIN mode (operation mode 3)

LIN-USART in LIN mode supports several additional features which reduces the interrupt load on the CPU.

- Detection of bus error
- Internal loop back feature
- Variable LIN break length generation

Detection of bus error

This feature is enabled by setting the bit USARTn_EFERH:DBE to '1'. The detection of bus error is done by enabling both transmission and reception, so that LIN node can read back its own transmission (as the LIN is single wire network). The physical bus error such as shorted to ground or Vcc can be found by comparing the value of transmitted and received data. If there is a difference between the transmitted and received value then USARTn_ESR:BUSERR flag is set. This results in an error interrupt if USARTn_EIER:BUSERRIE is set to '1'.

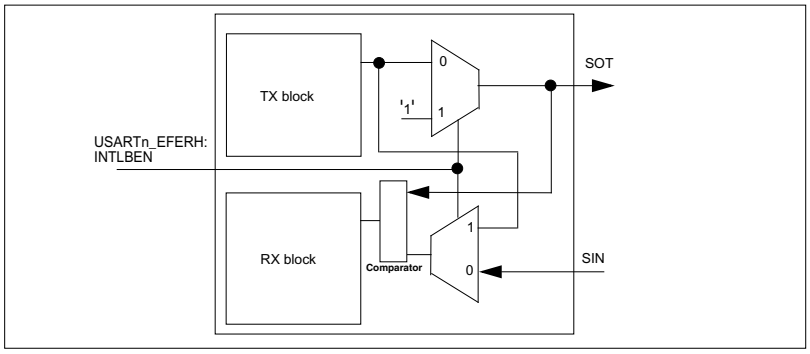
Note: Detection of bus error has to be disabled in internal loop back mode USARTn_EFERH:INTLBEN = '1' because in internal loop back mode SOT is always ‘1’.

Internal loop back feature

This feature is enabled by setting the bit USARTn_EFERH:INTLBEN to '1'.

By enabling this feature, the transmitter output is internally fed back to the receiver input so that the software can send and receive back the same data for the purpose of comparison. This can be used to reduce the tester time/debug efforts for testing LIN hardware.

Figure 31-67. Schematic for internal loop back feature



Variable length LIN break generation

LIN break of variable length from 13-20 bits can be generated. This is possible by setting the bits (USARTn_EFERH:LBL2, USARTn_ESCR:LBL[1:0]) to the required value.

31.8.3.2 Operation in asynchronous LIN mode (operation mode 3)

LIN-USART as LIN master

In LIN master mode the master determines the baud rate of the whole sub bus, therefore slave devices have to synchronize to the master. The desired baud rate remains fixed in master operation after initialization.

Writing a '1' into the USARTn_ECCR:LBR bit generates a 13 to 20-bit time low-level on the SOT pin, which is the LIN synchronization break and the start of a LIN message. Thereby the USARTn_SSR:TDRE flag goes '0' and is reset to '1' after the break, and generates a transmission interrupt for the CPU (if USARTn_SSR:TIE is '1'). The length of the synchronization break to be sent can be determined by the USARTn_EFER:LBL2 bit and USARTn_ESCR:LBL1/0 bits as follows:

Table 31-60. LIN break length

USARTn_ EFER: LBL2	USARTn_ ESCR: LBL1	USARTn_ ESCR: LBL0	Length of break
'0'	'0'	'0'	13-bit times
'0'	'1'	'0'	14-bit times
'0'	'0'	'1'	15-bit times
'0'	'1'	'1'	16-bit times
'1'	'0'	'0'	17-bit times
'1'	'0'	'1'	18-bit times
'1'	'1'	'0'	19-bit times
'1'	'1'	'1'	20-bit times

The sync field is sent as byte data of 0x55 after the LIN break. To prevent a transmission interrupt, the 0x55 can be written to the USARTn_TDR just after writing the '1' to the USARTn_ECCR:LBR bit, although the USARTn_SSR:TDRE flag is '0'. The internal transmission shifter waits until the LIN break has finished and shifts the USARTn_TDR value out afterwards. In this case no interrupt is generated after the LIN break and before the start bit of the sync field (0x55).

31.8.4 Direct access to serial pins

LIN-USART allows the user to directly access the transmission pin (SOT) or the reception pin (SIN).

LIN-USART direct pin access

The LIN-USART provides the ability for the software to access directly serial input or output pin. The software can always monitor the incoming serial data by reading USARTn_ESCR:SIOP bit. By setting the Serial Output Pin Direct Access Enable (USARTn_ESCR:SOPE) bit, the software can force the SOT pin to a desired value.

Note:

This access is only possible if the transmission shift register is empty (i. e. no transmission activity). In LIN mode this function can be used for reading back the own transmission and is used for error handling if something is physically wrong with the single wire LIN bus.

Write the desired value to USARTn_ESCSR:SIOPS (to set the output pin) and USARTn_ESCCR:SIOPC (to clear the output pin) before enabling the output pin direct access (USARTn_ESCR:SOPE) to prevent undesired output level because USARTn_ESCR:SIOP holds the last written value.

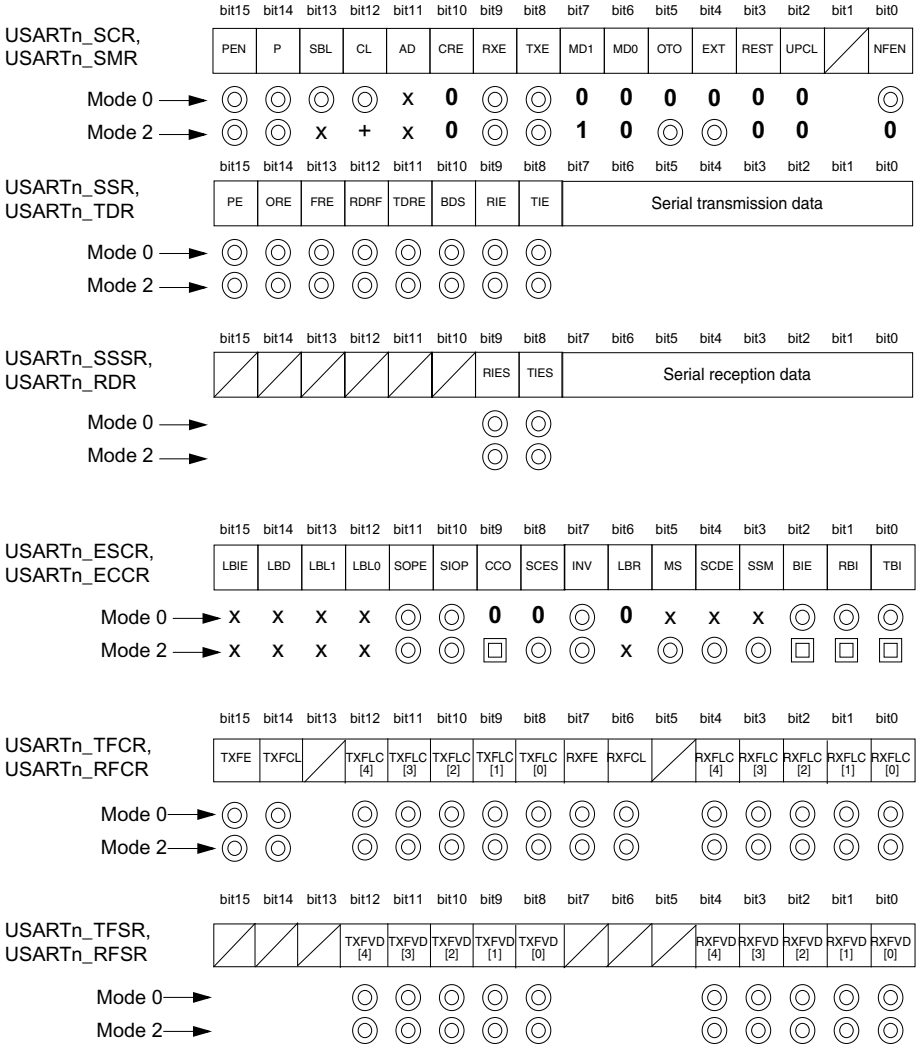
31.8.5 Bidirectional communication function (normal mode)

In operation mode 0 or 2, normal serial bidirectional communication is available. Select operation mode 0 for asynchronous communication and operation mode 2 for synchronous communication.

31.8.5.1 Bidirectional communication function

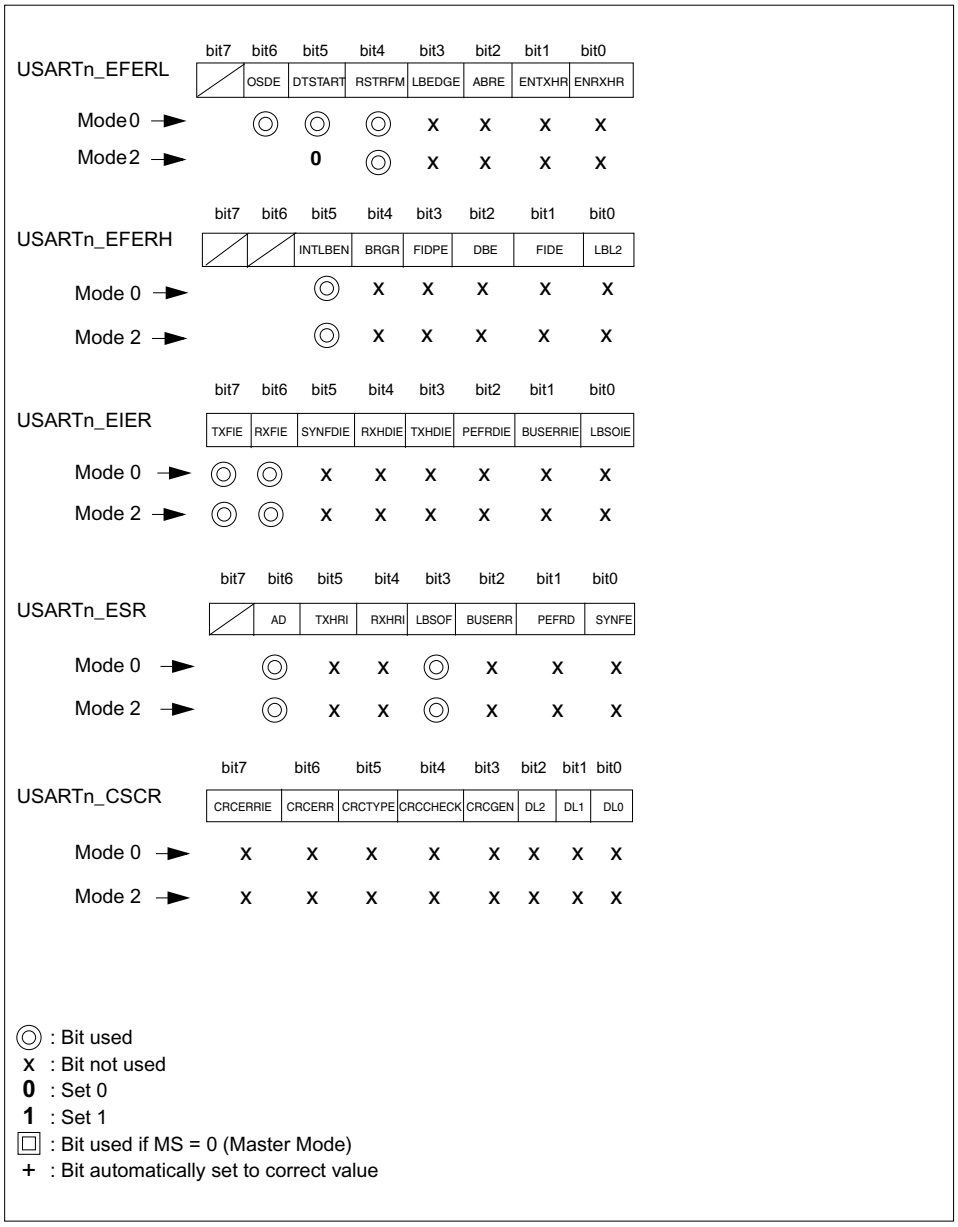
The settings shown in Figure 31-68 are required to operate LIN-USART in normal mode (operation mode 0 or 2).

Figure 31-68. Settings for LIN-USART operation Modes 0 and 2



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
□ : Bit used if MS = 0 (Master Mode)
+ : Bit automatically set to correct value

Figure 31-69. Settings for LIN-USART operation mode 0 and 2 (contd).



Inter-CPU connection

As shown in Figure 31-70, interconnect two CPUs in LIN-USART mode 2.

Figure 31-70. Connection example of LIN-USART mode 2 bidirectional communication

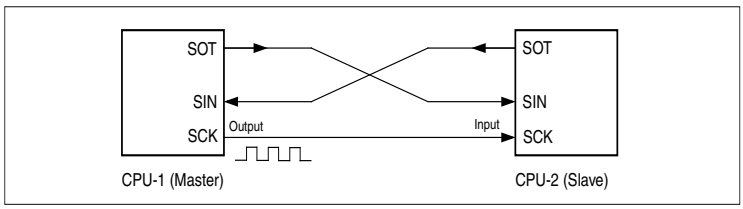
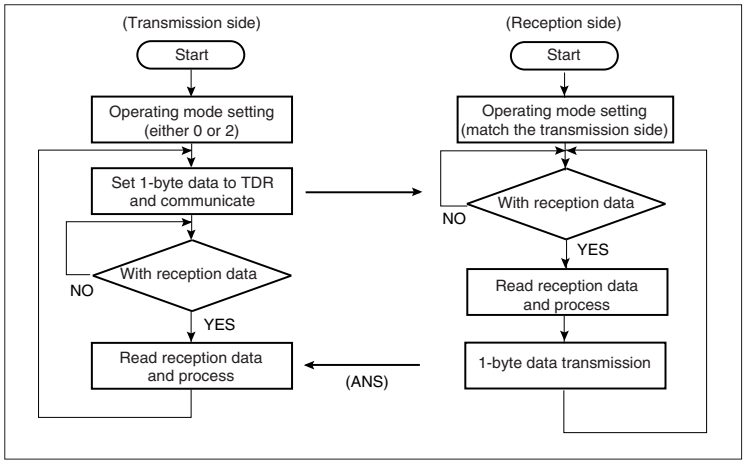


Figure 31-71. Example of master-slave communication flowchart



Note: In synchronous mode if the continuous clock output is not used, the master has to also send data if it wants to receive data in order to provide a serial clock to the slave.

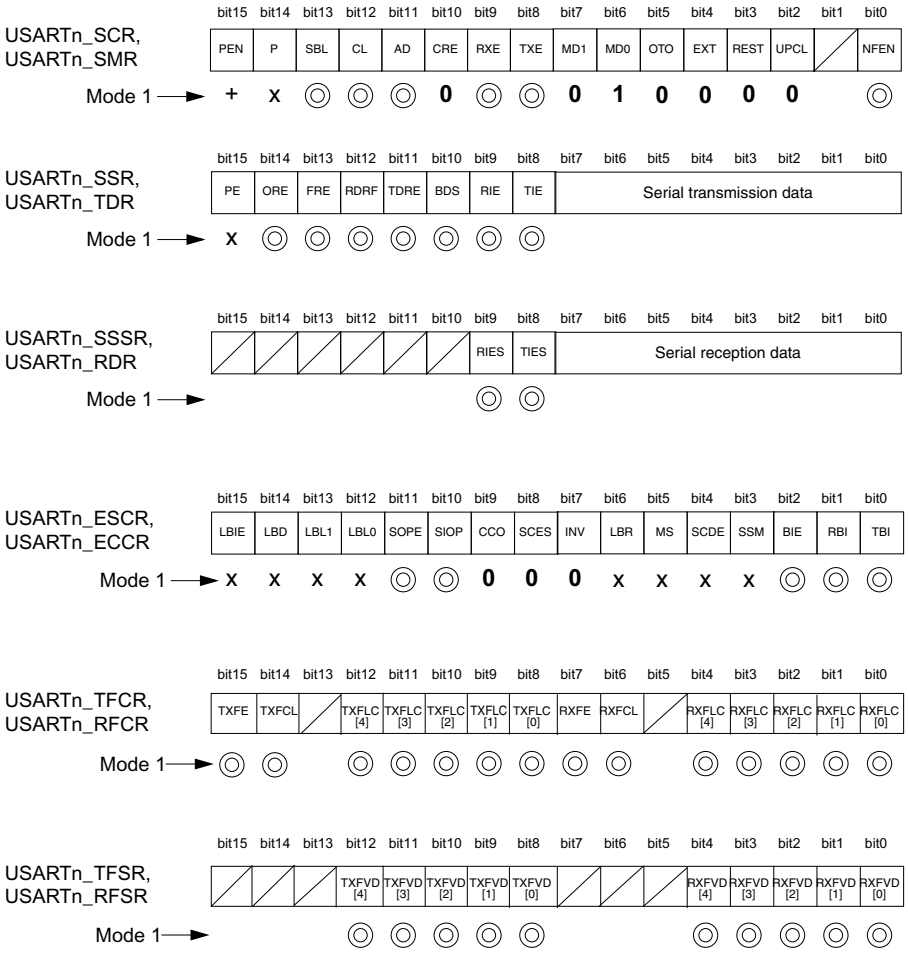
31.8.6 Master-slave communication function (multiprocessor mode)

LIN-USART communication with multiple CPUs connected in master-slave mode is available for both master or slave systems.

31.8.6.1 Master-slave communication function

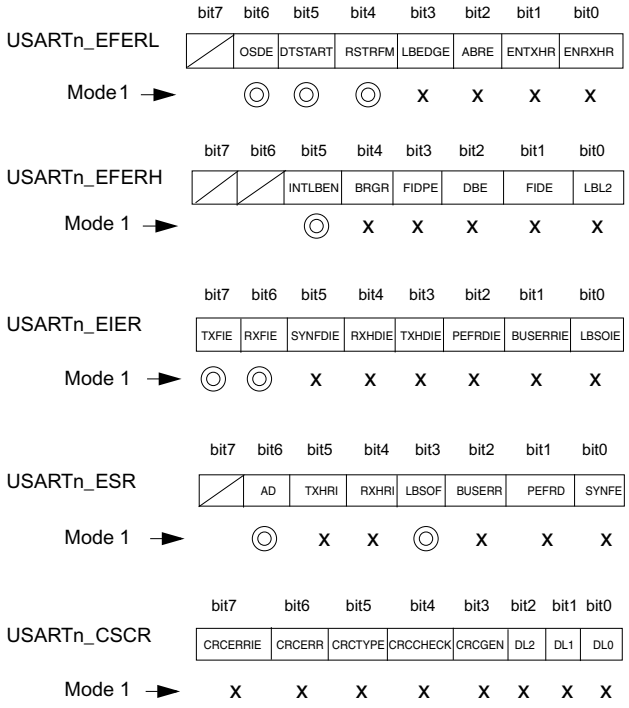
The settings shown in Figure 31-72 and Figure 31-73 are required to operate the LIN-USART in multiprocessor mode (operation mode 1).

Figure 31-72. Settings for LIN-USART operation mode 1



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
+ : Bit automatically set to correct value

Figure 31-73. Settings for LIN-USART operation mode 1

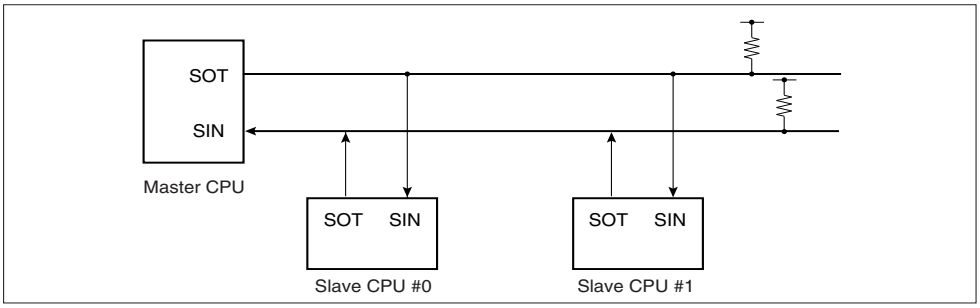


◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
+ : Bit automatically set to correct value

Inter-CPU connection

As shown in Figure 31-74, a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. LIN-USART can be used for the master or slave CPU.

Figure 31-74. Connection example of LIN-USART master-slave communication



Function selection

Select the operation mode and data transfer mode for master-slave communication as shown in Table 31-61.

Table 31-61. Selection of the master-slave communication function

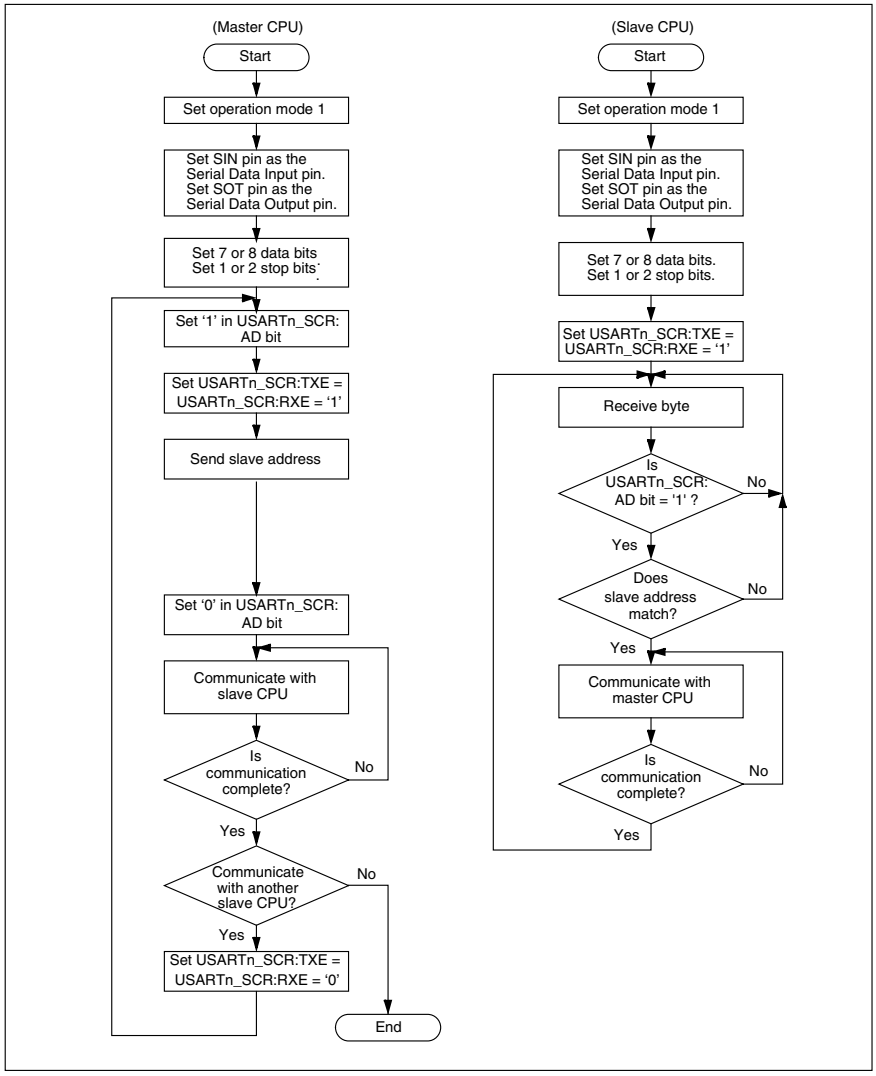
	Operation mode		Data	Parity	Synchronization method	Stop bit	Bit direction
	Master CPU	Slave CPU					
Address transmission and reception	Mode 1 (transmit/receive USARTn_SCR:AD bit)	Mode 1 (transmit/receive USARTn_SCR:AD bit)	USARTn_SCR: AD = '1' + 7- or 8-bit address	None	Asynchronous	1 or 2-bits	LSB or MSB first
Data transmission and reception			USARTn_SCR: AD = '0' + 7- or 8-bit data				

Communication procedure

When the master CPU transmits address data, communication starts. The USARTn_SCR:AD bit in the address data is set to '1', and the communication destination slave CPU is selected. Each slave CPU checks the address data using a program. When the address data indicates the address assigned to a slave CPU, the slave CPU communicates with the master CPU.

Figure 31-75 shows a flowchart of master-slave communication (multiprocessor mode).

Figure 31-75. Master-slave communication flowchart



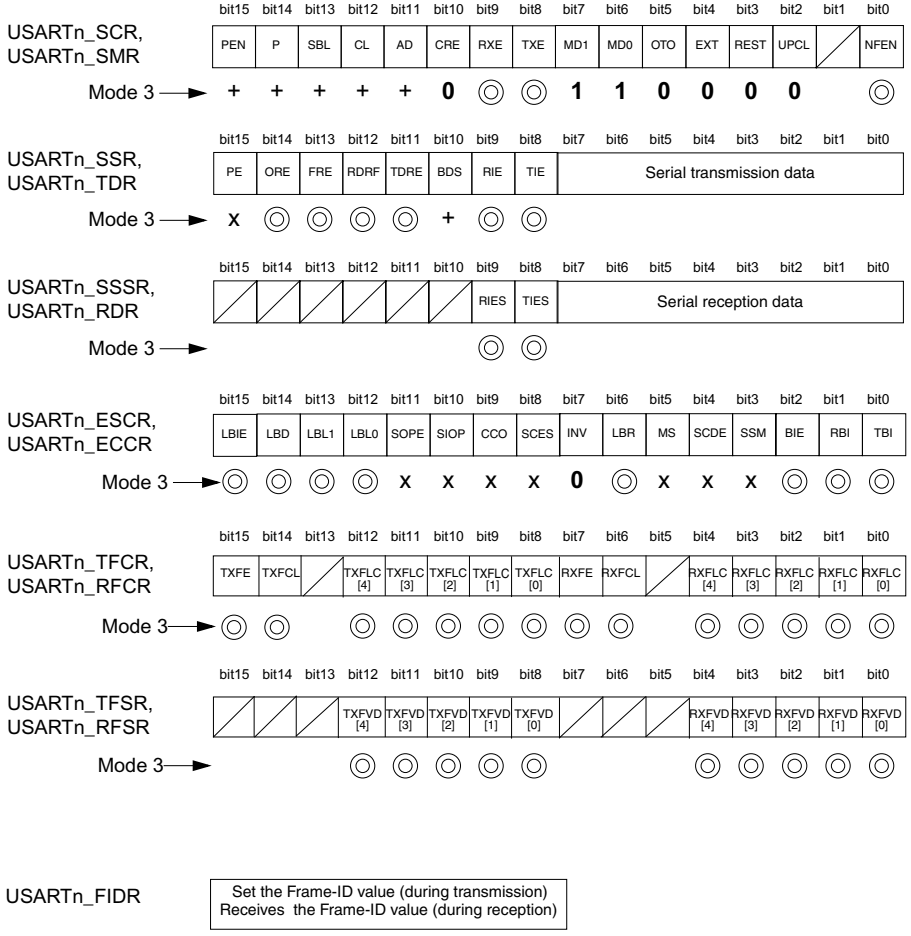
31.8.7 LIN communication function

LIN-USART communication with LIN devices is available for the LIN master.

31.8.7.1 LIN master communication function

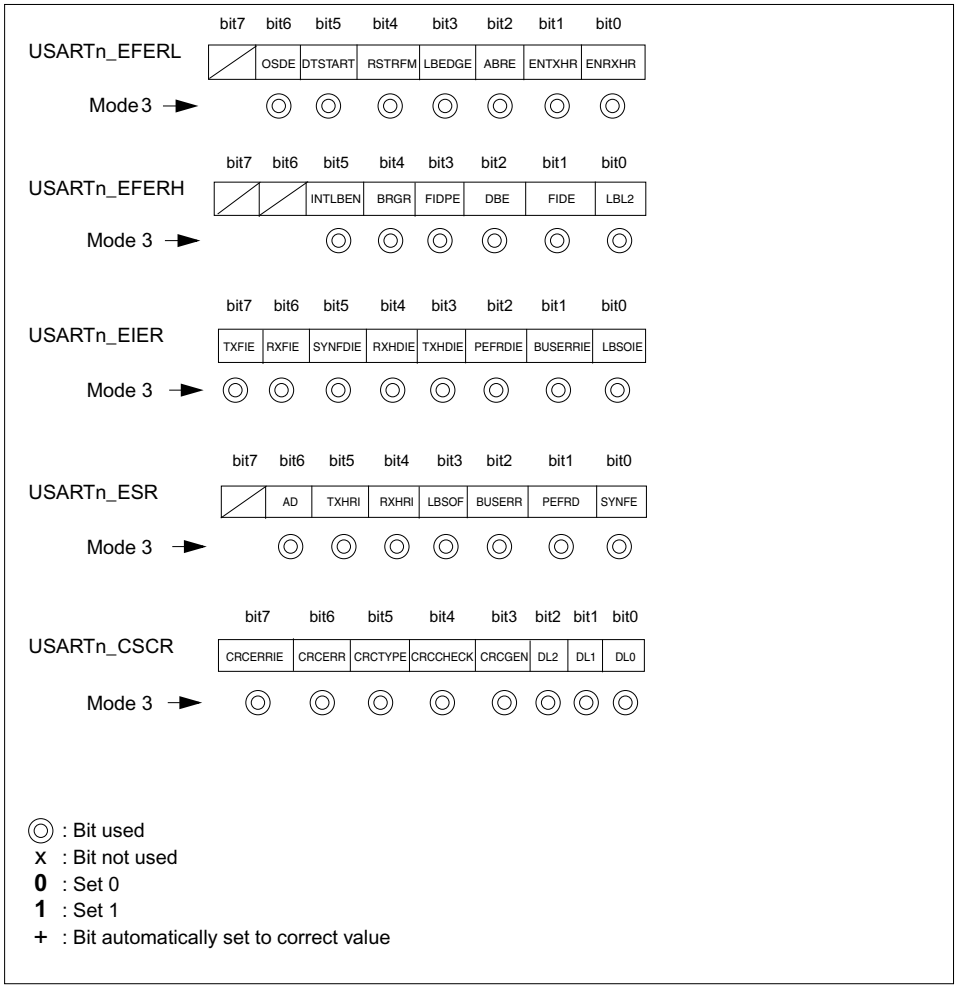
The settings shown in Figure 31-76 and Figure 31-77 are required to operate LIN-USART in LIN communication mode (operation mode 3).

Figure 31-76. Settings for LIN-USART in operation mode 3 (LIN)



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
+ : Bit automatically set to correct value

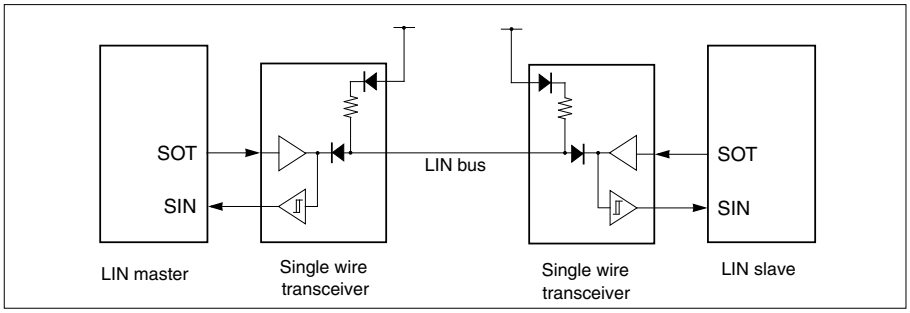
Figure 31-77. Settings for LIN-USART in operation mode 3 LIN



LIN device connection

As shown in Figure 31-78, a communication system of one LIN-master device and a LIN-slave device. LIN-USART can operate as a LIN-master.

Figure 31-78. Connection example of a small LIN-bus system

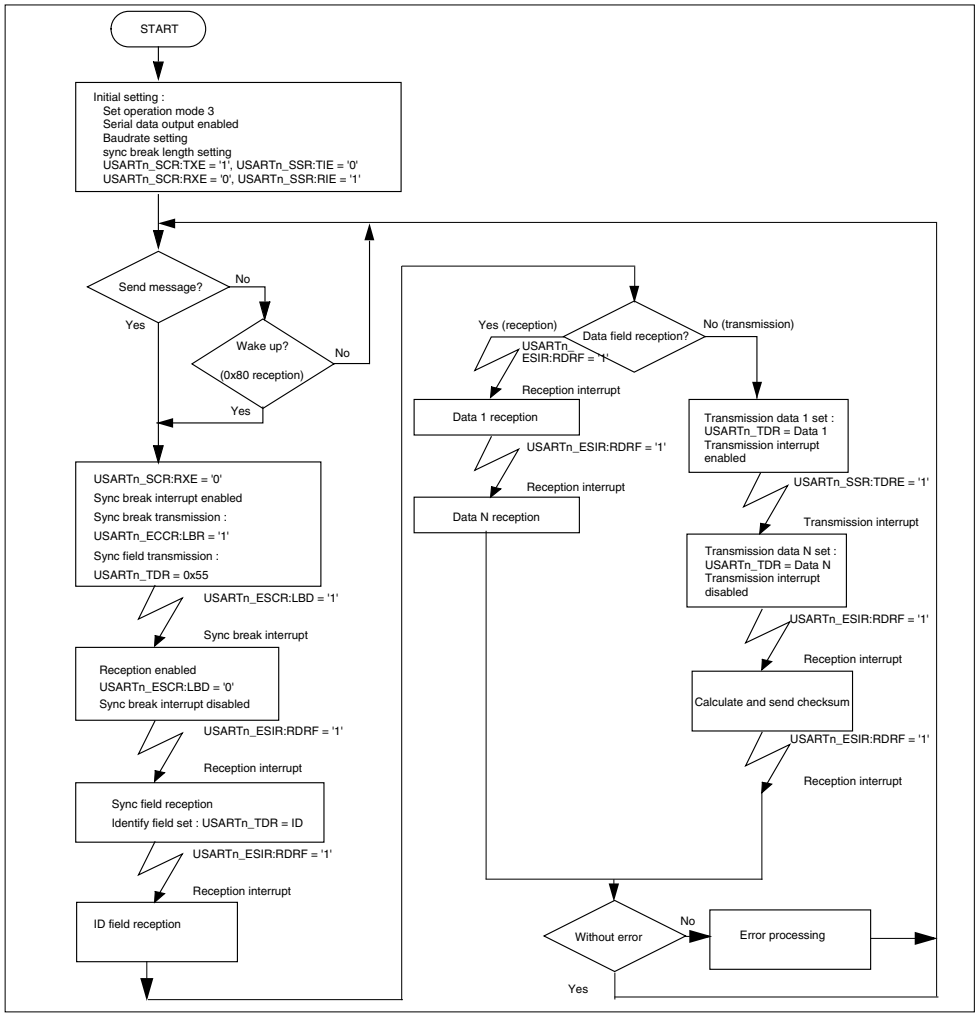


31.8.8 Sample flowcharts for LIN-USART in LIN communication (operation mode 3)

This section contains sample flowcharts for LIN-USART in LIN communication.

LIN-USART as master device

Figure 31-79. LIN-USART LIN master flowchart without automatic header function



31.9 Notes on using the LIN-USART

The following section describes notes on using LIN-USART.

Enabling operation

In LIN-USART, the control register (USARTn_SCR) has USARTn_SCR:TXE (transmission) and USARTn_SCR:RXE (reception) operation enable bits. Both, data transmission and reception operations, must be enabled before the communication starts because they have been disabled as the default value (initial value). The operation can also be canceled by disabling these bits. In synchronous slave mode (mode 2) the transmission enable (USARTn_SCR:TXE) and reception enable (USARTn_SCR:RXE) shall be set before the clock starts at the SCK pin.

Communication mode setting

Set the communication mode while the system is not operating. If the mode is changed during transmission or reception, the transmission or reception is stopped and possible data will be lost.

Transmission interrupt enabling timing

The default (initial value) of the Transmission Data Empty Flag bit (USARTn_SSR:TDRE) is '1' (no transmission data and transmission data write enable state). A transmission interrupt request is generated as soon as the transmission interrupt request is enabled (USARTn_SSR:TIE = '1'). Ensure to set the USARTn_SSR:TIE flag to '1' after setting the transmission data to avoid an immediate interrupt.

Interrupt enabling when FIFO is enabled

If RX FIFO is enabled (USARTn_RFCR:RXFE = '1'), USARTn_SSR:RIE has no effect on reception interrupt generation. Instead USARTn_EIER:RXFIE is used to enable or disable reception interrupt. Similarly if TX FIFO is enabled (USARTn_TFCR:TXFE = '1'), TIE has no effect on transmission interrupt generation. Instead USARTn_EIER:TXFIE is used to enable or disable transmission interrupt.

Using LIN operation mode 3

The LIN features are available in mode 3, but using mode 3 sets the LIN-USART data format automatically to LIN format (8N1, LSB first). Note: The length of the sync break for transmission is variable but for reception it is fixed 10.5-bit times.

Changing operation settings

It is strongly recommended to reset the LIN-USART after changing operation settings. Particularly if (for example) start-/stop-bits added to or removed from the data format. It is recommended to disable the communication (USARTn_SCR:RXE = '0', USARTn_SCR:TXE = '0'), if the LIN-USART setting or mode is changed or the LIN-USART is reset.

Using synchronous slave mode without continuous clock (USARTn_ESCR:CCO = ’0’)

In synchronous slave mode without continuous clock, the write to Transmission Data Register (USARTn_TDR) must be done before providing the clock for transmission operation. The approximate time which the data must be written into the USARTn_TDR, should be greater than half serial clock time period plus one CLKP time period (assuming that it takes one CLKP time period for data to be written to the TDR register).

Using transmission/reception FIFO

FIFO has to be cleared using USARTn_TFCR:TXFLC[4:0]/USARTn_RFCR:RXFLC[4:0] before enabling or disabling the respective FIFO.

Using last bit shift out interrupt

In all the modes, synchronization of status flag USARTn_ESR:LBSOF in bus bridge will add up to the interrupt latency, and also the ISR (Interrupt Service Routine) call will add up to the interrupt latency.

LIN slave settings

Set the baud rate before receiving the first LIN sync break for the slave operation. Otherwise, duration of the sync break cannot be correctly checked against the minimum requirement of the LIN specification (13 master bit time and 10.5 slave bit time).

Bus idle function

The bus idle function cannot be used in synchronous slave mode 2.

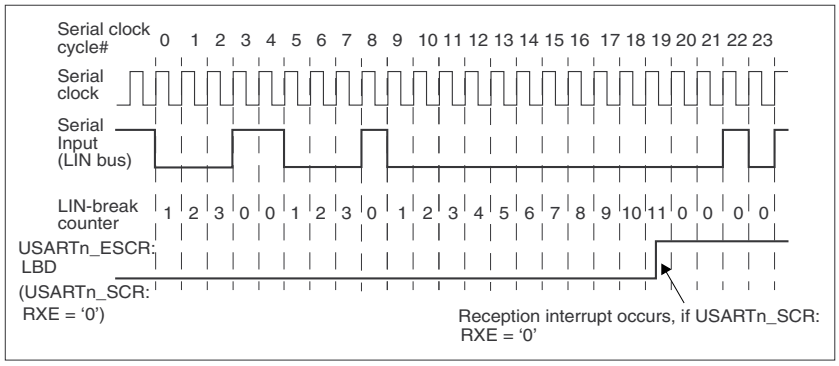
Clearing reception errors

Clearing reception errors resets the reception state machine when USARTn_EFER:RSTRFM = '0'. Therefore, check any reception errors before the next start bit or start condition is met, to not disturb any ongoing reception. If USARTn_EFER:RSTRFM = '1' reception state machine is not reset by USARTn_SCR:CRE.

LIN sync field wait state

In mode 3 (LIN operation), the USARTn_ESCR:LBD bit in the USARTn_ESCR register is set to '1' if the input signal is kept at '0' for more than or equal to 10.5-bit times. Then the LIN-USART waits for the following sync field to be received. If the LIN-USART is set into this state for other reasons than the sync break, it should be initialized by the software reset (USARTn_SMR:UPCL = '1'). In mode 3, LIN break detection is always working in the background and is level sensitive or edge sensitive depending on USARTn_EFER:LBEDGE. Be careful in case of a bus error (bus always dominant). The LIN break detection flag (USARTn_ESCR:LBD) will go '1' or stay '1' after each 10.5 bit times when USARTn_EFER:LBEDGE = '0' independent from enabled or disabled LIN-Break Detection interrupt. If LIN break detection interrupt is used, check and always clear this flag in the reception interrupt handler. If USARTn_EFER:LBEDGE is '0' (level sensitive detection of LIN break start) LIN break detection is restarted with every high level on SIN till a valid LIN sync field has been detected. If USARTn_EFER:LBEDGE is '1' (edge sensitive detection of LIN break start) LIN break detection is restarted with every falling edge on SIN till a valid LIN sync field has been detected. The following figure shows the behavior of the LIN break detection counter. This figure does not distinguish between USARTn_EFER:LBEDGE settings because the behaviour in this scenario is same.

Figure 31-80. LIN synch break detection



Effects of reception errors and USARTn_SCR:CRE bit

- If USARTn_EFERL:RSTRFM = '0', bit USARTn_SCR:CRE resets the reception state machine and next falling edge at SINn starts reception of new byte. Therefore, either set USARTn_SCR:CRE bit immediately (within half bit time) after receiving errors to prevent data stream de-synchronization or wait an application dependent time after receiving errors and set USARTn_SCR:CRE, when SINn is idle.
- If USARTn_EFERL:RSTRFM = '1' and USARTn_SCR:CRE is set, the reception state machine is not reset.

The startup sequence using Synchronous Slave mode

- In Synchronous slave mode (mode 2) the transmission enable bit (USARTn_SCR:TXE) and reception enable bit (USARTn_SCR:RXE) shall be set before the clock starts at the SCK port to avoid an invalid data output.

Figure 31-81. Timing of the USARTn_SCR:CRE bit (USARTn_EFER:RSTRFM = ’0’)

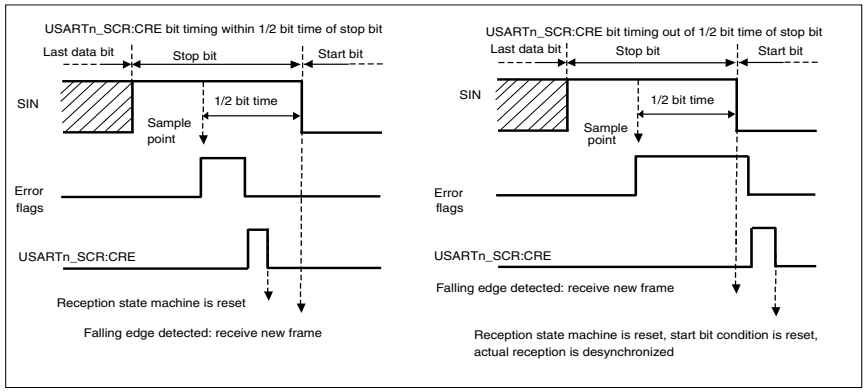
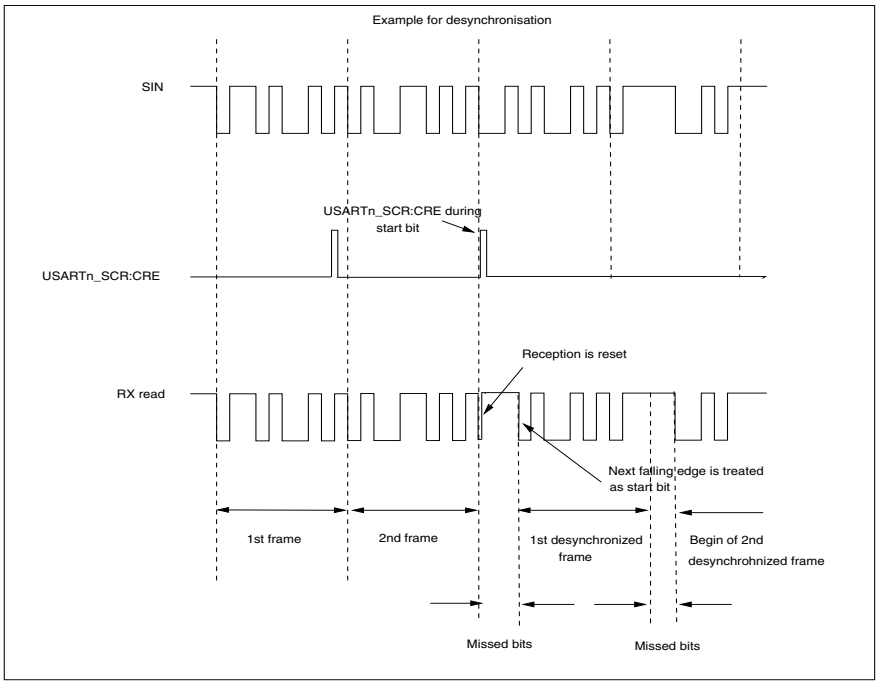


Figure 31-82. Data stream de-synchronization example (USARTn_EFERL:RSTRFM = ’0’)



Start bit detection

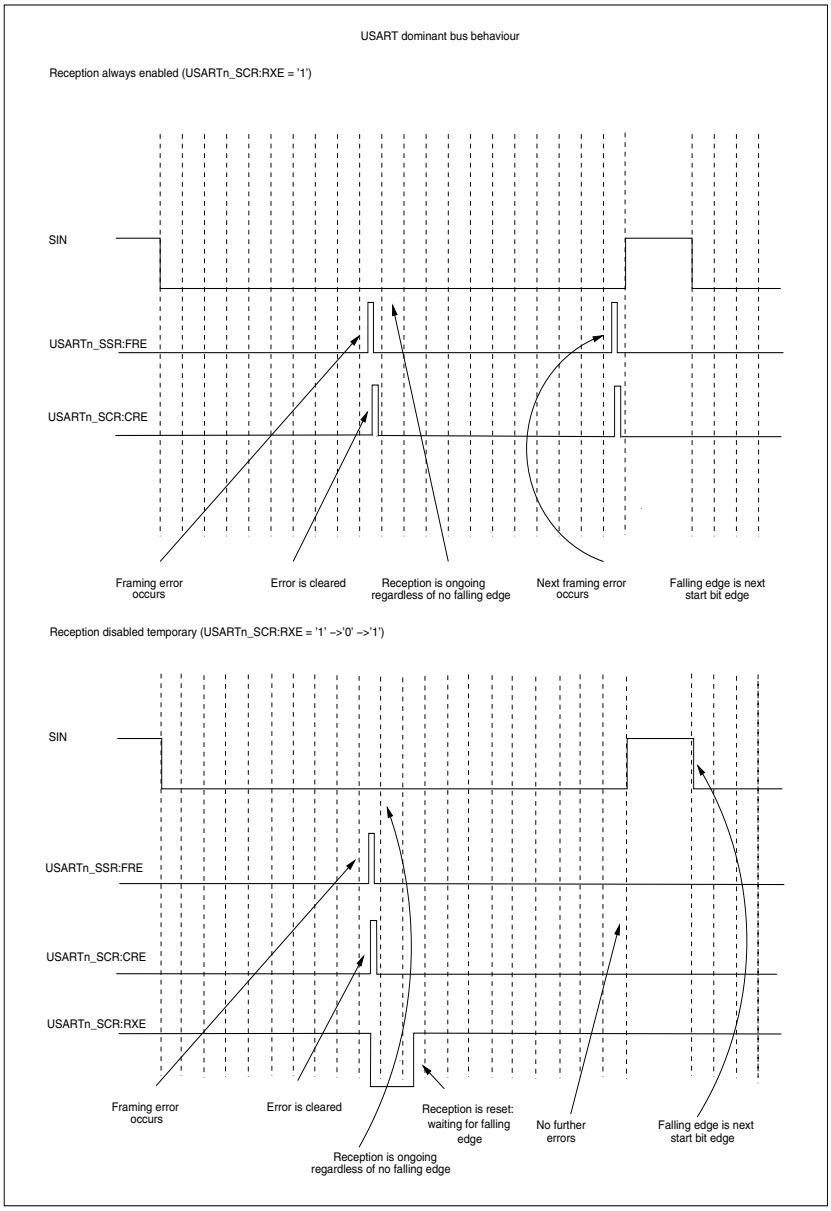
If USARTn_EFERL:DTSTART = '0'

In case a framing error occurred (stop bit: USARTn_SIN = '0') and the next start bit (USARTn_SIN = '0') follows immediately, this start bit is recognized regardless of no falling edge before. This is used to keep the LIN-USART synchronized to the data stream and to determine bus always dominant errors (see Figure 31-83 upper figure) by producing next framing errors, if a recessive stop bit is expected. If this behaviour is not wanted, disable the reception temporarily (USARTn_SCR:RXE = 1 -> 0 -> 1) after framing error. In this case, reception goes on at next falling edge on USARTn_SIN. (See Figure 31-83 lower figure).

If USARTn_EFERL:DTSTART = '1':

Next falling edge of SIN input after USARTn_SSR:FRE is considered as valid start bit.

Figure 31-83. LIN-USART dominant bus behaviour (USARTn_EFERL:DTSTART = ’0’)



Debug mode

LIN-USART enters into debug mode when USARTn_DEBUG:DBGEN is set to '1' and DEBUG input is high. In this mode, transmission/reception and baud rate generator is stopped after completing the ongoing byte transmission/reception.

In mode 3 (LIN mode), even if DEBUG input is asserted in any of the header or response fields (e.g LIN break, sync field, data bytes) during transmission/reception, the LIN-USART completes the transmission of the ongoing field or the transmission/reception of the current data byte before going into debug mode.

In modes 0, 1, and 2, if DEBUG input is asserted in the middle of transmission/reception, the LIN- USART completes the transmission/reception of the current byte before going into debug mode. When USARTn_DEBUG:DBGEN is set to '1' and DEBUG input is high when LIN-USART is in mode 2 (synchronous slave mode), the external clock provided through SCKn pin is masked. When USARTn_RDR is read, USARTn_SSR:RDRF is cleared. If RX-FIFO is enabled, the RX- FIFO pointer is

changed to point to the next valid data. However, when data are read in debug mode, the same data cannot no longer be accessed by applications. The data provided through the SINn pin is also masked.

When the DEBUG input is deasserted or when USARTn_DEBUG:DBGEN is cleared in debug mode, LIN moves back to its previous state from which it entered into the debug state.

32. 400 kHz I²C Interface

This section describes the function and operation of the fast I²C Interface.

Note: Do not use the I²C Interface as a master in a multi-master system.

32.1 Overview of I²C Interface

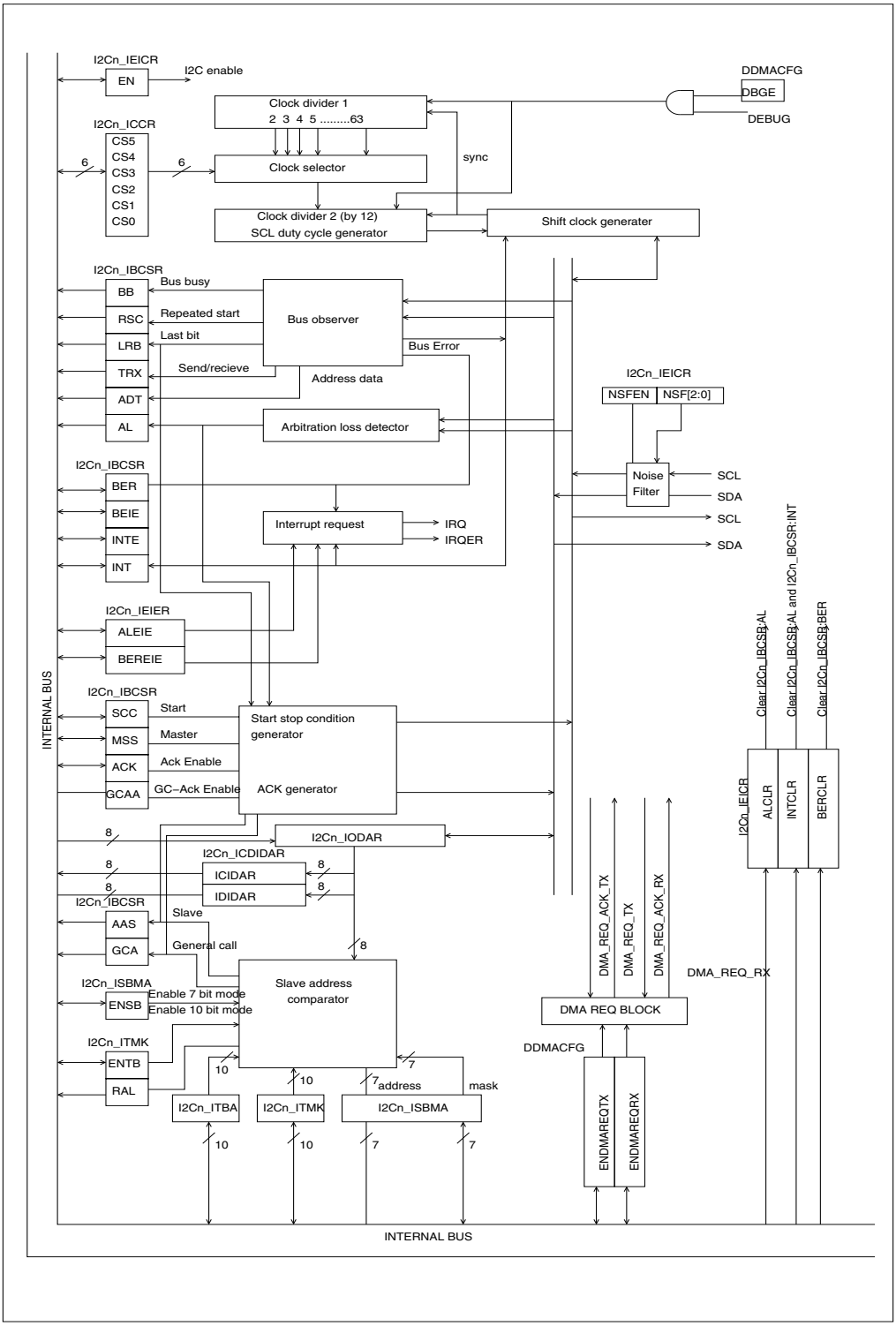
This section describes the features and the block diagram of the I²C Interface. Features of I²C Interface The I²C Interface is a serial I/O port supporting the Inter IC bus and operating as a master/slave device on the I²C bus. The I²C bus is a multi-master bus. More than one device capable of controlling the bus can be connected to it. Data on the I²C bus can be transferred at a rate of up to 100 kbps in standard mode or up to 400 kbps in fast mode. The number of interfaces connected to the bus is solely dependent on the bus capacitance limit of 400 pF.

Only two bus lines are required, a Serial Data Line (SDA) and a Serial Clock Line (SCL). Each device connected to the bus is software addressable by a unique address, and simple master/slave relationships exist at all times. Masters can operate as master-transmitters or as master-receivers. On-chip filtering rejects spikes on the bus data line to preserve data integrity. The features of the I²C Interface are:

- Master/slave transmitting and receiving functions
- Clock synchronisation function
- General call addressing support
- Transfer direction detection function
- Repeated start condition generation and detection function
- Bus error detection function
- Seven-bit addressing as master and slave
- Ten-bit addressing as master and slave
- Possibility to give the interface a 7- and a 10-bit slave address
- Acknowledgement of slave address reception can be disabled (master-only operation)
- Address masking to give the interface several slave addresses (in 7- and 10-bit mode)
- Up to 400 kbps transfer rate
- Possibility to use built-in noise filters for SDA and SCL
- It can receive data at 400 kbps if the peripheral clock is higher than 6 MHz regardless of the prescaler setting
- It can generate Microcontroller Unit (MCU) interrupts on transmission and bus error events
- It supports slow-down by a slave on a bit and byte level
- It provides support for MCU debug mode
- It provodes support for the DMA interface

The I²C Interface does not support SCL clock stretching on a bit level since it can receive the full 400 kbps data rate if the peripheral clock is higher than 6 MHz regardless of the prescaler setting. However, clock stretching on a byte level is performed since SCL is pulled low during an interrupt (I2Cn_IBCSR:INT = '1').

Figure 32-1. Block diagram of I²C Interface



32.2 I²C Interface registers

This section describes the function of the I²C Interface registers.

The suffix 'n' in the register name indicates that the register is in instance 'n' of the module.

Registers of I²C Interface

The following registers are available for each instance of I²C Interface

- Bus Control and Status Register (I2Cn_IBCSR)
- Ten Bit Slave Address Register (I2Cn_ITBA)
- Ten Bit Slave Address Mask Register (I2Cn_ITMK)
- Seven Bit Slave Mask and Address Register (I2Cn_ISBMA)
- Output Data Register (I2Cn_IODAR)
- Clock Control Register (I2Cn_ICCR)
- CPU and DMA Input Data Register (I2Cn_ICDIDAR)
- Interface Enable and Interrupt Clear Register (I2Cn_IEICR)
- Debug and DMA Configuration Register(I2Cn_DDMACFG)
- Error Interrupt Enable Register (I2Cn_IEIER)

Memory layout of I²C Interface registers

Table 32-1. Memory layout of I²C Interface registers

Offset	+1	+0
0x00000000	I2Cn_IBCSR 00000000 00000000	
0x00000002	I2Cn_ITBA 00000000 00000000	
0x00000004	I2Cn_ITMK 00XXXX11 11111111	
0x00000006	I2Cn_ISBMA 01111111 00000000	
0x00000008	reserved XXXXXXXX	I2Cn_IODAR 00000000
0x0000000A	reserved XXXXXXXX	I2Cn_ICCR 00111111
0x0000000C	I2Cn_ICDIDAR 00000000 00000000	
0x0000000E	I2Cn_IEICR 00000000 00000000	
0x00000010	I2Cn_DDMACFG 00000000 00000000	
0x00000012	reserved XXXXXXXX	I2Cn_IEIER 00000000

32.2.1 Bus Control and Status Register (I2Cn_IBCSR)

The Bus Control and Status Register (I2Cn_IBCSR) has the following functions:

- Bus busy detection
- Repeated start condition detection
- Arbitration loss detection
- Acknowledge detection
- Data transfer direction indication
- Addressing detection as slave
- General call address detection
- Address/data detection
- Interrupt enabling flags
- Interrupt generation flag
- Bus error detection flag
- Repeated start condition generation
- Master/slave mode selection
- General call acknowledge generation enabling
- Data byte acknowledge generation enabling

Bus Control and Status Register (I2Cn_IBCSR)

The lower byte of the I2Cn_IBCSR register is read-only and all bits are controlled by the hardware. All bits (lower byte) are cleared if the interface is not enabled (i.e. I2Cn_IEICR:EN = '0'). Write access to the I2Cn_IBCSR register should only occur while the I2Cn_IBCSR:INT = '1' or if a transfer is to be started.

Note:

The user should not write to this register during an ongoing transfer since changes to the I2Cn_IBCSR:ACK or I2Cn_IBCSR:GCAA bits could result in bus errors.

All bits in this register except the I2Cn_IBCSR:BER and I2Cn_IBCSR:BEIE bits are cleared if the interface is not enabled (i.e. I2Cn_IEICR:EN = '0').

Figure 32-2. Bus Control and Status Register (I2Cn_IBCSR)

I2Cn_IBCSR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BER	BEIE	SCC	MSS	ACK	GCAA	INTE	INT	BB	RSC	AL	LRB	TRX	AAS	GCA	ADT
Rp	RpWp	Rp0Wp1	RpWp	RpWp	RpWp	RpWp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-2. Bus Control and Status Register (I2Cn_IBCSR) bits

Bit Position	Bit Field Name	Bit Description
[15]	BER	Bus Error bit The bus error interrupt flag is set by the hardware and cleared by the user. It is cleared by writing '1' to I2Cn_IEICR:BERCLR. '0': No bus error detected '1': One of the following error conditions has been detected ▫ Start or stop conditions have been detected at the wrong places, for example during an address data transfer or during the transfer of bits 2 to 11 (ACK). ▫ A 10-bit address header with read access is received before a 10-bit write access. When this bit is set:: ■ The I2Cn_IEICR:EN bit is cleared ■ The I^2C interface goes to pause status ■ Data transfer is interrupted ■ All bits in the I2Cn_IBCSR and the I2Cn_IBCSR registers except I2Cn_IBCSR:BER and I2Cn_IBCSR:BEIE are cleared The I2Cn_IBCSR:BER bit must be cleared before the interface can be re-enabled.
[14]	BEIE	Bus Error Interrupt Enable bit This bit enables a bus error interrupt. It can only be changed by the user. '0': Bus error interrupt disabled '1': Bus error interrupt enabled Setting this bit to '1' enables the generation of an interrupt (IRQ) when the I2Cn_IBCSR:BER bit is set to '1'.
[13]	SCC	Start Condition Continue bit This write-only bit is used to generate a repeated start condition. Reading it always returns '0'. Write access: '0': No effect '1': Generate repeated start condition during master transfer A repeated start condition is generated if '1' is written to this bit, while an interrupt in master mode (i.e. I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1') and the I2Cn_IBCSR:INT bit is cleared automatically.

Table 32-2. Bus Control and Status Register (I2Cn_IBCSR) bits

Bit Position	Bit Field Name	Bit Description
[12]	MSS	Master Slave Select bit The master/slave mode selection bit can only be set by the user but cleared by the user and the hardware. '0': Go to slave mode '1': Go to master mode, generate a start condition and write the address data byte to the I2Cn_IODAR register. It is cleared if an arbitration loss event occurs as the master is sending If '0' is written to this bit during a master interrupt (i.e. when I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1'), the I2Cn_IBCSR:INT bit is cleared automatically, a stop condition will be generated and the data transfer ends. **Note:** The I2Cn_IBCSR:MSS bit is reset immediately; the generation of the stop condition can be checked by polling the I2Cn_IBCSR:BB bit. If '1' is written to this bit while the bus is idle (i.e. when I2Cn_IBCSR:MSS = '0' and I2Cn_IBCSR:BB = '0'), a start condition is generated and the contents of the I2Cn_IODAR:IODAR register (which should contain the address data) is sent. If '1' is written to the I2Cn_IBCSR:MSS bit while the bus is in use (i.e. when I2Cn_IBCSR:BB = '1', I2Cn_IBCSR:TRX = '0' and I2Cn_IBCSR:MSS = '0'), the interface waits until the bus is free before it starts sending. If, in the meantime, the interface is addressed as slave with write access (data reception), it will start sending data when the bus is again free. If during this time the interface sends data as a slave (i.e. when I2Cn_IBCSR:AAS = '1' and I2Cn_IBCSR:TRX = '1'), it must wait until the bus is free again. It is important to check whether the interface was addressed as a slave (I2Cn_IBCSR:AAS = '1'), and whether it sent the data byte successfully (I2Cn_IBCSR:MSS = '1') or unsuccessfully (I2Cn_IBCSR:AL = '1') at the next interrupt.

Table 32-2. Bus Control and Status Register (I2Cn_IBCSR) bits

Bit Position	Bit Field Name	Bit Description
[11]	ACK	Acknowledge bit This bit enables acknowledge generation on data byte reception. It only can be changed by the user. '0': The interface will not acknowledge on data byte reception '1': The interface will acknowledge on data byte reception This bit is not valid when receiving address byte in slave mode if the interface detects its seven or ten bit slave address, it will acknowledge if the corresponding enable bit (I2Cn_ITMK:ENTB or I2Cn_ISBMA:ENSB) is set. Write access to this bit should occur during an interrupt (I2Cn_IBCSR:INT = '1') or if the bus is idle (I2Cn_IBCSR:BB = '0'). In addition the interface must be enabled (I2Cn_IEICR:EN = '1') and there has to be no bus error (I2Cn_IBCSR:BER = '0').
[10]	GCAA	General Call Address Acknowledge bit This bit enables the generation of an acknowledgement by the interface when a general call address has been received. It can only be changed by the user. '0': The interface will not acknowledge on receipt of a general call address byte '1': The interface will acknowledge on receipt of a general call address byte Write access to this bit should occur during an interrupt (I2Cn_IBCSR:INT = '1') or if the bus is idle (I2Cn_IBCSR:BB = '0'). In addition, the interface must be enabled (I2Cn_IEICR:EN = '1') and there can be no bus error (I2Cn_IBCSR:BER = '0').
[9]	INTE	Interrupt Enable bit This bit enables the generation of an interrupt. It can only be changed by the user. '0': Interrupt disabled '1': Interrupt enabled Setting this bit to '1' enables interrupt generation (IRQ) when the I2Cn_IBCSR:INT bit is set to '1' (by the hardware).

Table 32-2. Bus Control and Status Register (I2Cn_IBCSR) bits

Bit Position	Bit Field Name	Bit Description
[8]	INT	Interrupt Flag bit This bit is set by the hardware and cleared by the user. It is cleared by writing '1' to I2Cn_IEICR:INTCLR. '0': Transfer has not ended or the interface is not involved in current transfer, or the bus is idle '1': Set at the end of a one byte data transfer or reception, including the ACK bit, under the following conditions: □ Device is bus master. □ Device is addressed as a slave. □ General call address received. □ Arbitration loss occurred. Set when the address data has been received (i.e. set after the first byte if a 7-bit address has been received or after the second byte if a 10-bit address has been received), including the ACK bit, if the device is addressed as a slave While INT is '1', the SCL line will hold a level 'L' signal. Clearing this bit releases the SCL line and either executes the transfer of the next byte or generates a repeated start or stop condition. Additionally, this bit is cleared if a '1' is written to the I2Cn_IBCSR:SCC bit or the I2Cn_IBCSR:MSS bit is cleared. This bit is also cleared when I2Cn_DDMACFG:DMAMODE is set, and a read operation at I2Cn_ICDIDAR:IDIDAR or a write operation at I2Cn_IODAR is performed.
[7]	BB	Bus Busy bit This bit indicates the status of the I2C bus. '0': Stop condition detected (bus idle) '1': Start condition detected This bit is set to '1' if a start condition has been detected. It is reset upon a stop condition.
[6]	RSC	Repeated Start Condition bit This bit indicates the detection of a repeated start condition. '0': Repeated start condition not detected '1': Repeated start condition detected (bus in use) This bit is cleared at the end of an address data transfer (i.e. I2Cn_IBCSR:ADT = '0') or on detection of a stop condition.

Table 32-2. Bus Control and Status Register (I2Cn_IBCSR) bits

Bit Position	Bit Field Name	Bit Description
[5]	AL	Arbitration Lost bit This bit indicates an arbitration loss. '0': No arbitration loss detected '1': Arbitration loss occurred during master sending This bit is cleared by writing '1' to either the I2Cn_IEICR:INT-CLR, I2Cn_IBCSR:MSS or I2Cn_IEICR:ALCLR bits. An arbitration loss occurs if: ■ The data sent does not match the data read on the SDA line at the rising edge of the SCL. ■ A repeated start condition is generated by another master in the first bit of a data byte. ■ The interface could not generate a start or stop condition because of a signal transition from '1' to '0' by a particular external condition observed at the SCL line.
[4]	LRB	Last Received Bit This bit is used to indicate the acknowledgement from the receiving device. '0': Receiver has acknowledged '1': Receiver has not acknowledged It is changed by the hardware upon receipt of the I2Cn_IBCSR:ACK bit and is also cleared by a start or stop condition.
[3]	TRX	Transferring Data bit This bit indicates the status of the data transmission operation. '0': Not transmitting data '1': Transmitting data It is set to '1': ■ If a start condition was generated in master mode. ■ If addressed as slave in read access. It is set to '0' if: ■ The bus is idle (I2Cn_IBCSR:BB = '0'). ■ An arbitration loss has occurred. ■ '1' is written to the I2Cn_IBCSR:SCC bit during a master interrupt (I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1'). ■ The I2Cn_IBCSR:MSS bit has been cleared during a master interrupt (I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1'). ■ The interface is in slave mode and the last transferred byte was not acknowledged. ■ The interface is in slave mode and is receiving data. ■ The interface is in master mode and is reading data from a slave.

Table 32-2. Bus Control and Status Register (I2Cn_IBCSR) bits

Bit Position	Bit Field Name	Bit Description
[2]	AAS	Addressed As Slave bit This bit indicates the detection of slave addressing. '0': Not addressed as slave '1': Addressed as slave This bit is cleared by a (repeated) start or stop condition. It is set if the interface detects its 7- and/or 10-bit slave address.
[1]	GCA	General Call Address bit This bit indicates the detection of a general call address (0x00). '0': General call address is not received as slave '1': General call address is received as slave This bit is cleared by a (repeated) start or stop condition.
[0]	ADT	Address Data Transfer bit This bit indicates the detection of an address data transfer. '0': Incoming data is not address data (or bus is not in use) '1': Incoming data is address data This bit is set to '1' by a start condition. It is cleared after the second byte if a 10-bit slave address header with write access is detected, otherwise it is cleared after the first byte. This bit is also cleared: ■ When '0' is written to the I2Cn_IBCSR:MSS bit during a master interrupt (I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1'). ■ When '1' is written to the I2Cn_IBCSR:SCC bit during a master interrupt (I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1'). ■ When The I2Cn_IBCSR:INT bit is cleared. ■ At the beginning of every byte transfer if the interface is not involved in the current transfer as master or slave.

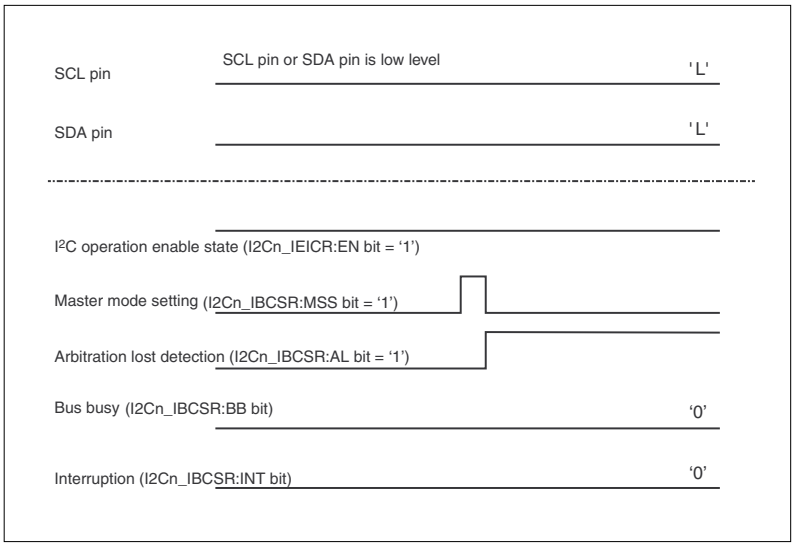
I2Cn_IBCSR:SCC and I2Cn_IBCSR:MSS bit competition

Repeated start condition generation and stop condition generation—When a '1' is written to the I2Cn_IBCSR:SCC bit and a '0' to the I2Cn_IBCSR:MSS bit simultaneously, I2Cn_IBCSR:MSS bit clearing takes priority. A stop condition is generated and the interface enters slave mode. If specific conditions are met, the I2Cn_IBCSR:AL (arbitration lost) bit does not set the I2Cn_IBCSR:INT (interrupt) bit. These conditions are presented in Figure 32-3 and Figure 32-4.

Case 1: When the SCL and SDA signals are kept at 'L':

The I2Cn_IBCSR:AL bit is set immediately after the I2Cn_IBCSR:MSS bit set to '1' while the I2Cn_IBCSR:BB bit indicates '0' (no start condition is detected). However, the I2Cn_IBCSR:AL bit will not set the I2Cn_IBCSR:INT bit under this circumstance.

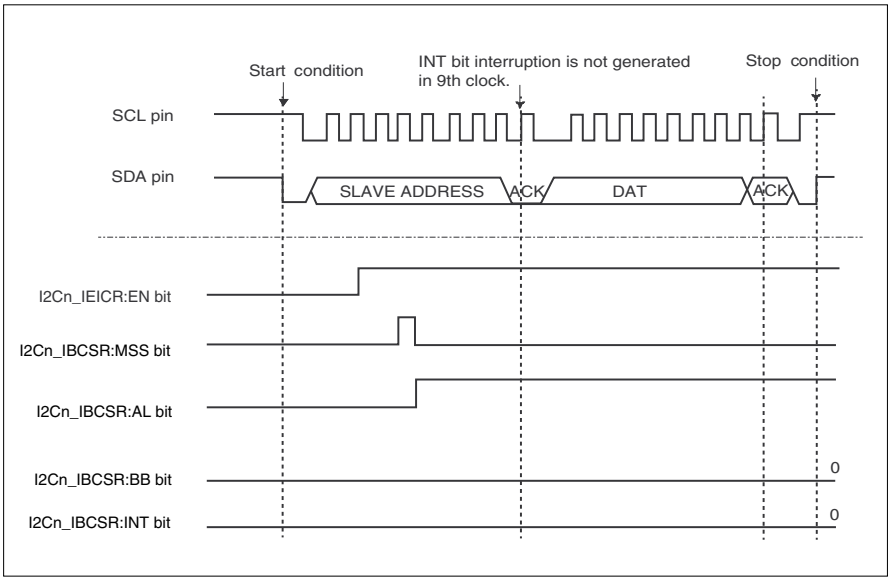
Figure 32-3. Timing diagram which illustrates that an interrupt does not occur upon detection of I2Cn_IBCSR:AL bit = '1'



Case 2: When the I²C Interface is enabled while there is ongoing communication with another bus master:

The interface participates in the I²C bus while the bus is occupied with ongoing communication if the I2Cn_IEICR:EN bit is set from '0' to '1'. In this case the I2Cn_IBCSR:BB bit stays '0' (no start condition is detected) and setting the I2Cn_IBCSR:MSS bit to '1' results in the I2Cn_IBCSR:AL bit indicating '1'. However, the I2Cn_IBCSR:AL bit will not set the I2Cn_IBCSR:INT bit under this circumstance.

Figure 32-4. Timing diagram which illustrates that an interrupt does not occur upon detection of I2Cn_IBCSR:AL bit = '1'



If a symptom as described above can occur, follow the procedure below for software processing:

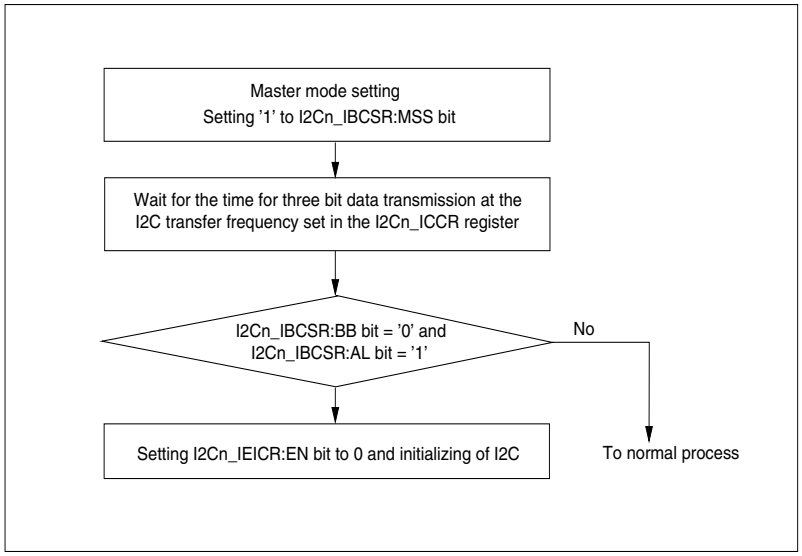
1. Execute the instruction that generates a start condition (i.e. set the I2Cn_IBCSR:MSS bit to '1').
2. Use, for example, the timer function to wait for the time it takes for 3-bit data transmission at the I²C transfer frequency, which is set in the I2Cn_ICCR:CS register[a].

Example: Time for three-bit data transmission at an I^2C transfer rate of
100 kbps = (1 / (100 x 10^3)) x 3 = 30 µs

3. If I2Cn_IBCSR:AL = '1' and I2Cn_IBCSR:BB = '0', then set the I2Cn_IEICR:EN bit to '0' to initialize the I^2C unit. If I2Cn_IBCSR:AL and I2Cn_IBCSR:BB have different values, then perform normal processing.

A sample flow is given below.

Figure 32-5. I^2C arbitration flow diagram

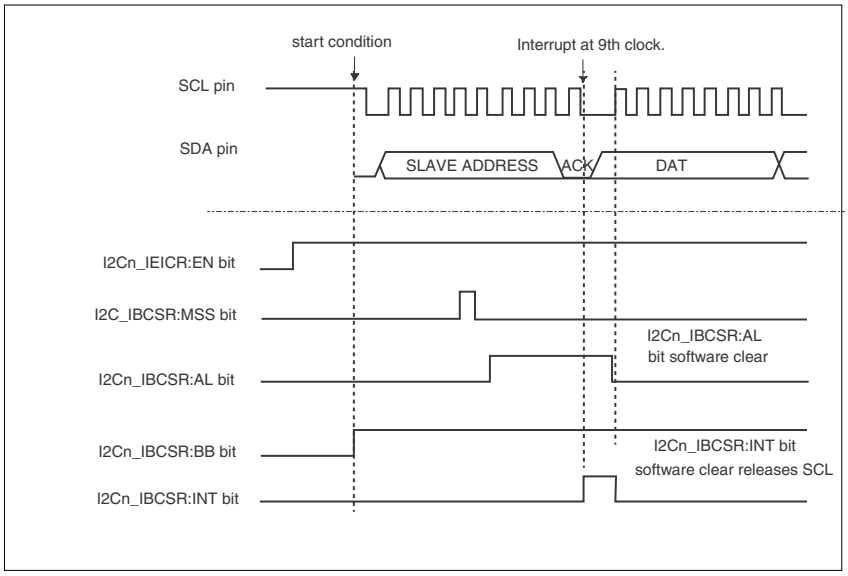


a: Arbitration lost is detected within 3-bit time after setting the I2Cn_IBCSR:MSS bit to '1'.

Example of the occurrence of an interrupt (i.e. I2Cn_IBCSR:INT bit = '1') upon detection of 'I2Cn_IBCSR:AL bit = '1'.

With reference to Figure 32-6, when an instruction which generates a start condition is executed (i.e. setting the I2Cn_IBCSR:MSS bit to '1') with 'bus busy' detected (i.e. I2Cn_IBCSR:BB bit = '1') and arbitration is lost, the I2Cn_IBCSR:INT bit interrupt occurs when I2Cn_IBCSR:AL bit = '1' had been detected.

Figure 32-6. Timing diagram which illustrates that an interrupt is generated upon detection of I2Cn_IBCSR:AL bit = '1'



32.2.2 Ten Bit Slave Address Register (I2Cn_ITBA)

The Ten Bit Slave Address Register (I2Cn_ITBA) designates the ten bit slave address.

Ten Bit Slave Address Register (I2Cn_ITBA)

Write access to this register is only possible if the interface is disabled (I2Cn_IEICR:EN = '0').

Figure 32-7. Ten Bit Slave Address Register (I2Cn_ITBA)

I2Cn_ITBA															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	TA[9]	TA[8]	TA[7]	TA[6]	TA[5]	TA[4]	TA[3]	TA[2]	TA[1]	TA[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-3. Ten Bit Slave Address Register (I2Cn_ITBA) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	read0	-
[9:0]	TA	Ten Bit Slave Address When address data is received in slave mode, it is compared to the I2Cn_ITBA register if the 10-bit address is enabled (i.e. I2Cn_ITMK:ENTB = '1'). An acknowledgement is sent to the master after a 10-bit address header with write access[1] has been received. The second incoming byte is then compared to the I2Cn_ITBA register. If a match is detected, an acknowledgement is sent to the master device and the I2Cn_IBCSR:AAS bit is set. Additionally, the interface issues an acknowledgement upon the receipt of a 10-bit header with read access[2] after a repeated start condition. All bits of the slave address may be masked using the I2Cn_ITMK register. The 10-bit slave address received is written back to the I2Cn_ITBA register - it is only valid while the I2Cn_IBCSR:AAS bit is '1'.

1. A 10-bit header (write access) consists of the following bit sequence: 1 1 1 1 0 TA[9] TA[8] 0
2. A 10-bit header (read access) consists of the following bit sequence: 1 1 1 1 0 TA[9] TA[8] 1

32.2.3 Ten Bit Slave Address Mask Register (I2Cn_ITMK)

The Ten Bit Slave Address Mask Register (I2Cn_ITMK) contains the 10-bit slave address mask and the 10-bit slave address enable bit.

Ten Bit Slave Address Mask Register (I2Cn_ITMK)

Figure 32-8. Ten Bit Slave Address Mask Register (I2Cn_ITMK)

I2Cn_ITMK															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ENTB	RAL	reserved	reserved	reserved	reserved	TM[9]	TM[8]	TM[7]	TM[6]	TM[5]	TM[4]	TM[3]	TM[2]	TM[1]	TM[0]
RpWp	Rp	-	-	-	-	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1

Table 32-4. Ten Bit Slave Address Mask Register (I2Cn_ITMK) bits

Bit Position	Bit Field Name	Bit Description
[15]	ENTB	Enable Ten Bit (10-bit) Slave Address This bit enables the 10-bit slave address (and the acknowledgement generated when it has been received). Write access to this bit is only possible if the interface is disabled (i.e. I2Cn_IEICR:EN = '0'). '0': 10-bit address disabled '1': 10-bit address enabled
[14]	RAL	Received Slave Address Length bit This bit indicates whether the interface was addressed as a 7- or 10-bit slave. It is a read-only bit. '0': Addressed as 7-bit slave '1': Addressed as 10-bit slave This bit can be used to determine whether the interface was addressed as a 7- or 10-bit slave if both slave addresses are enabled (i.e. I2Cn_ITMK:ENTB = '1' and I2Cn_ISBMA:ENSB = '1'). Its contents are only valid if the I2Cn_IBCSR:AAS bit is '1'. This bit is also reset if the interface is disabled (I2Cn_IEICR:EN = '0').
[13:10]	reserved	-

Table 32-4. Ten Bit Slave Address Mask Register (I2Cn_ITMK) bits

Bit Position	Bit Field Name	Bit Description
[9:0]	TM	Ten Bit (10-bit) Slave Address Mask bits This register is used to mask the 10-bit slave address of the interface. Write access to these bits is only possible if the interface is disabled (i.e. I2Cn_IEICR:EN = '0'). '0': Bit is not used in slave address comparison '1': Bit is used in slave address comparison This can be used to make the interface acknowledge on multiple 10-bit slave addresses. Only the bits set to '1' in this register are used in the 10-bit slave address comparison. The slave address received is written back to the I2Cn_ITBA register and therefore may be determined by reading the I2Cn_ITBA register if the I2Cn_IBCSR:AAS bit is '1'. **Note:** If the address mask is changed after the interface has been enabled, the slave address should also be set again since it could have been overwritten by a previously received slave address.

32.2.4 Seven Bit Slave Mask and Address Register (I2Cn_ISBMA)

The Seven Bit Slave Mask and Address Register (I2Cn_ISBMA) contains the 7-bit slave address, the 7-bit slave address mask and the 7-bit slave address enable bit.

Seven Bit Slave Mask and Address Register (I2Cn_ISBMA)

Write access to I2Cn_ISBMA register is only possible if the interface is disabled (i.e. I2Cn_IEICR:EN = '0').

Figure 32-9. Seven Bit Slave Mask and Address Register (I2Cn_ISBMA)

I2Cn_ISBMA															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ENSB	SM[6]	SM[5]	SM[4]	SM[3]	SM[2]	SM[1]	SM[0]	read0	SA[6]	SA[5]	SA[4]	SA[3]	SA[2]	SA[1]	SA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Table 32-5. Seven Bit Slave Mask and Address Register (I2Cn_ISBMA) bits

Bit Position	Bit Field Name	Bit Description
[15]	ENSB	Enable Seven Bit (7-bit) Slave Address bits This bit enables the 7-bit slave address (and the acknowledgement generated after it has been received). '0': 7-bit slave address disabled '1': 7-bit slave address enabled
[14:8]	SM	Seven Bit (7-bit) Slave Address Mask bits This register is used to mask the 7-bit slave address of the interface. '0': Bit is not used in slave address comparison '1': Bit is used in slave address comparison This can be used to make the interface acknowledge on multiple 7-bit slave addresses. Only the bits set to '1' in this register are used in the 7-bit slave address comparison. The received slave address is written back to the I2Cn_ISBMA:SA[6:0] register and therefore may be determined by reading the I2Cn_ISBMA:SA[6:0] register if the I2Cn_IBCSR:AAS bit is '1'. **Note:** If the address mask is changed after the interface has been enabled, the slave address should also be set again since it could have been overwritten by a previously received slave address.
[7]	read0	-

Table 32-5. Seven Bit Slave Mask and Address Register (I2Cn_ISBMA) bits

Bit Position	Bit Field Name	Bit Description
[6:0]	SA	Seven Bit (7-bit) Slave Address bit When address data is received in slave mode, it is compared to these 7 bits, SA[6:0] if the 7-bit address is enabled (i.e. I2Cn_ISBMA:ENSB = '1'). If a match is detected, an acknowledge signal is sent to the master device and the I2Cn_IBCSR:AAS bit is set. All slave address bits may be masked using the I2Cn_ISBMA:SM[6:0] register. The 7-bit slave address received is written back to SA[6:0]; it is only valid while the I2Cn_IBCSR:AAS bit is '1'. The interface does not compare the contents of this register to the incoming data if a 10-bit header or a general call is received.

32.2.5 Output Data Register (I2Cn_IODAR)

The Output Data Register for the 400 kHz I^2C Interface is used by the CPU to send data.

Output Data Register (I2Cn_IODAR)

Figure 32-10. Output Data Register (I2Cn_IODAR)

I2Cn_IODAR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
IODAR[7]	IODAR[6]	IODAR[5]	IODAR[4]	IODAR[3]	IODAR[2]	IODAR[1]	IODAR[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 32-6. Output Data Register (I2Cn_IODAR) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	IODAR	Output Data Register This register is used in serial data transfers and transfers MSB data first. It is double buffered on the write side so that when the bus is in use (i.e. I2Cn_IBCSR:BB = '1'), write data can be loaded to the register for serial transfer. The data byte is loaded into the internal transfer register if the I2Cn_IBCSR:INT bit in the I2Cn_IBCSR register is being cleared or the bus is idle (i.e. I2Cn_IBCSR:BB = '0'). Writing this register clears the I2Cn_IBCSR:INT bit if the I2Cn_DDMACFG:DMAMODE bit is set.

32.2.6 Clock Control Register (I2Cn_ICCR)

The Clock Control Register is used to configure the prescaler.

Clock Control Register (I2Cn_ICCR)

Figure 32-11. Clock Control Register (I2Cn_ICCR)

I2Cn_ICCR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	CS[5]	CS[4]	CS[3]	CS[2]	CS[1]	CS[0]
Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	1	1	1	1	1	1

Table 32-7. Clock Control Register (I2Cn_ICCR) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5:0]	CS	Clock Prescaler bits These bits select the serial bit rate. They can only be changed if the interface is disabled (i.e. I2Cn_IEICR:EN = '0'). For more description refer to the section 'Clock prescaler settings'.

Clock prescaler settings

The calculation formula for I2Cn_ICCR:CS[5:0] is as follows:

$$\text{Bitrate} = \frac{\phi}{n*12 + 16} \quad n>0$$

ϕ: peripheral clock, noise filter disabled

$$\text{Bitrate} = \frac{\phi}{n*12 + (16+y)} \quad n>0$$

ϕ: peripheral clock, noise filter enabled
y: varies from 1 to 5 according to noise filter configuration:
y = 1 for I2Cn_IEICR:NSF[2:0] = ’000’
y = 2 for I2Cn_IEICR:NSF[2:0] = ’001’, ’010’
y = 3 for I2Cn_IEICR:NSF[2:0] = ’011’, ’100’
y = 4 for I2Cn_IEICR:NSF[2:0] = ’101’, ’110’
y = 5 for I2Cn_IEICR:NSF[2:0] = ’111’

Table 32-8. Prescaler settings

n	I2Cn_ICCR: CS[5]	I2Cn_ICCR: CS[4]	I2Cn_ICCR: CS[3]	I2Cn_ICCR: CS[2]	I2Cn_ICCR: CS[1]	I2Cn_ICCR: CS[0]
1	'0'	'0'	'0'	'0'	'0'	'1'
2	'0'	'0'	'0'	'0'	'1'	'0'
3	'0'	'0'	'0'	'0'	'1'	'1'
...						
63	'1'	'1'	'1'	'1'	'1'	'1'

Note: Do not use n = '0' prescaler setting, it violates SDA/SCL timings.

Common peripheral clock frequencies

Table 32-9 shows the most common peripheral clock frequencies with their prescaler settings and the resulting sending bit rate:

Table 32-9. common peripheral clock frequencies

Peripheral clock frequency [MHz]	100 kbps (noise filter disabled)		400 kbps (noise filter enabled)			
	n	Bit rate [kbps]	n	Bit rate [kbps]	I2Cn_IEICR: NSF[2:0]	y
64	52	100	12	390.24	110	4
60	49	99.34	11	394.74	101	4
56	46	98.59	10	400	101	4
52	42	100	10	371.43	101	4
48	39	99.17	9	377.95	100	3
44	36	98.21	8	382.61	100	3
40	32	100	7	388.35	011	3
36	29	98.9	6	395.6	011	3
32	26	97.56	6	351.65	011	3
28	22	100	5	358.97	010	2
24	19	98.36	4	363.64	010	2
20	16	96.15	3	370.37	001	2
16	12	100	2	380.95	001	2
12	9	96.77	1	400	001	2
8	6	90.91	1	275.86	000	1

32.2.7 CPU and DMA Input Data Register (I2Cn_ICDIDAR)

The CPU and DMA Input Data Register for the 400 kHz I^2C Interface (I2Cn_ICDIDAR) are used to receive data by the CPU and DMA.

CPU and DMA Input Data Register (ICDIDAR)

Figure 32-12. CPU and DMA Input Data Register (I2Cn_ICDIDAR)

I2Cn_ICDIDAR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ICIDAR[7]	ICIDAR[6]	ICIDAR[5]	ICIDAR[4]	ICIDAR[3]	ICIDAR[2]	ICIDAR[1]	ICIDAR[0]	IDIDAR[7]	IDIDAR[6]	IDIDAR[5]	IDIDAR[4]	IDIDAR[3]	IDIDAR[2]	IDIDAR[1]	IDIDAR[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-10. CPU and DMA Input Data Register (I2Cn_ICDIDAR) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	ICIDAR	I^2C CPU Input Data Register By reading this register, the internal transfer register is read directly. Therefore data values received in this register are valid only if I2Cn_IBCSR:INT = '1'.
[7:0]	IDIDAR	I^2C DMA Input Data Register By reading the IDIDAR register, the internal transfer register is read directly; data values received in this register are valid only if I2Cn_IBCSR:INT = '1'. Reading this register clears the I2Cn_IBCSR:INT bit if I2Cn_DDMACFG:DMAMODE bit is set. This register can be read by enabling PPU access, otherwise reading causes a PPU error. However, reading this register by the debug master (DAP) will never clear the INT flag (irrespective of value of I2Cn_DDMACFG:DMAMODE).

32.2.8 Interface Enable and Interrupt Clear Register (I2Cn_IEICR)

The Interface Enable and Clear Register for the 400 kHz I^2C Interface (I2Cn_IEICR) is used to enable I^2C Interface operation, configure noise filter for I^2C and clear interrupt request flags.

Interface Enable and Interrupt Clear Register (I2Cn_IEICR)

Figure 32-13. Interface Enable and Interrupt Clear Register (I2Cn_IEICR)

I2Cn_IEICR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	NSF[2]	NSF[1]	NSF[0]	NSFEN	EN	read0	read0	read0	read0	read0	ALCLR	BERCLR	INTCLR
Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-11. Interface Enable and Interrupt Clear Register (I2Cn_IEICR) bits

Bit Position	Bit Field Name	Bit Description
[15:13]	read0	-

Table 32-11. Interface Enable and Interrupt Clear Register (I2Cn_IEICR) bits

Bit Position	Bit Field Name	Bit Description
[12:10]	NSF	Noise Filter Configuration bits These bits configure the maximum pulse width of single spikes on the SDA and SCL inputs that are suppressed by the built-in noise filter. '000': Suppress single spikes with a maximum pulse width of 0.5 peripheral clock cycles '001': Suppress single spikes with a maximum pulse width of 1 peripheral clock cycle '010': Suppress single spikes with a maximum pulse width of 1.5 peripheral clock cycles '011: Suppress single spikes with a maximum pulse width of 2 peripheral clock cycles '100': Suppress single spikes with a maximum pulse width of 2.5 peripheral clock cycles '101': Suppress single spikes with a maximum pulse width of 3 peripheral clock cycles '110': Suppress single spikes with a maximum pulse width of 3.5 peripheral clock cycles '111': Suppress single spikes with a maximum pulse width of 4 peripheral clock cycles The I2Cn_IEICR:NSF configuration is applicable when I2Cn_IEICR:NSFEN is set.
[9]	NSFEN	Noise Filter Enable bit '0': Noise filter disabled '1': Noise filter enabled This bit enables the noise filters for the SDA and SCL inputs. The noise filter will suppress single spikes according to the I2Cn_IEICR:NSF configuration. The NSFEN bit should be set to '1' if the interface is transmitting or receiving at data rates above 100 kbps.

Table 32-11. Interface Enable and Interrupt Clear Register (I2Cn_IEICR) bits

Bit Position	Bit Field Name	Bit Description
[8]	EN	Enable bit This bit enables the I^2C interface operation. It can only be set by the user but may be cleared by the user and the hardware. '0': Interface disabled '1': Interface enabled When this bit is set to '0' all bits in the I2Cn_IBCSR register (except the I2Cn_IBCSR:BER and I2Cn_IBCSR:BEIE bits) are cleared. In addition, the module is disabled and the I^2C lines are left open. It is cleared by the hardware if a bus error occurs (i.e. I2Cn_IBCSR:BER = '1'). **Note:** The interface immediately stops transmitting or receiving if it is being disabled. This might leave the I^2C bus in an undesired state.
[7:3]	read0	-
[2]	ALCLR	Arbitration Lost Interrupt Flag Clear bit This bit clears the arbitration lost interrupt request flag. '0': No effect '1': Clears the I2Cn_IBCSR:AL register bit Reading this bit always returns '0'.
[1]	BERCLR	Bus Error Interrupt Flag Clear bit This bit clears the bus error interrupt request flag. '0': No effect '1': Clears the I2Cn_IBCSR:BER register bit Reading this bit always returns '0'.
[0]	INTCLR	Interrupt Request Flag Clear bit This bit clears the interrupt request flag. '0': No effect '1': Clears the I2Cn_IBCSR:INT and I2Cn_IBCSR:AL register bits Reading this bit always returns '0'.

32.2.9 Debug and DMA Configuration Register(I2Cn_DDMACFG)

The Debug and DMA Configuration Register (I2Cn_DDMACFG) contains bits to enable/disable DMA request and the debug mode.

Debug and DMA Configuration Register (I2Cn_DDMACFG)

Figure 32-14. Debug and DMA Configuration Register (I2Cn_DDMACFG)

I2Cn_DDMACFG															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	DBGE	read0	read0	read0	read0	read0	DMAMODE	ENDMAREQTX	ENDMAREQRX
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-12. Debug and DMA Configuration Register (I2Cn_DDMACFG) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	DBGE	Debug Enable This bit is used to enable/disable the debug mode for I^2C. '0': Debug mode disabled '1': Debug mode enabled When DBGE is set to '1' and the processor is in debug state, both interrupt lines are masked if they have not been asserted already. In Master mode, SCL is Interrupts that have been masked during this condition are propagated again when the processor leaves debug state or DBGE is set to '0'. For the definition of debug state, refer to section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[7:3]	read0	-

Table 32-12. Debug and DMA Configuration Register (I2Cn_DDMACFG) bits

Bit Position	Bit Field Name	Bit Description
[2]	DMAMODE	DMA Mode for Data Registers This bit is used to enable/disable DMA mode for I2Cn_IODAR:IODAR and I2Cn_ICDIDAR:IDIDAR registers. '0': I2Cn_IBCSR:INT bit is not cleared with a write operation to I2Cn_IODAR:IODAR or a read operation from I2Cn_ICDIDAR:IDIDAR '1': I2Cn_IBCSR:INT bit is cleared with a write operation to I2Cn_IODAR:IODAR or read operation from I2Cn_ICDIDAR:IDIDAR
[1]	ENDMAREQTX	DMA Enable bit for Transmission This bit is used to enable/disable the DMA request for transmission. '0': DMA request for transmission is disabled '1': DMA request for transmission is enabled: TX DMA request is set if I2Cn_IBCSR:INT is set, I2Cn_IBCSR:TRX = '1' and I2Cn_IBCSR:AL = '0'. TX DMA request is cleared when it is acknowledged by the DMA controller.
[0]	ENDMAREQRX	DMA Enable bit for Reception This bit is used to enable/disable the DMA request for reception. '0': DMA request for reception is disabled '1': DMA request for reception is enabled: RX DMA request is set if I2Cn_IBCSR:INT is set, I2Cn_IBCSR:TRX = '0' and I2Cn_IBCSR:AL = '0'. RX DMA request is cleared when it is acknowledged by the DMA controller.

Note: The request generation behaviour is explained in details for different modes of operation:

1. Master mode as transmitter - the DMA request for transmission is generated after completing the transmission of data (address or data).
2. Master mode as receiver - the DMA request for transmission is generated after completing the transmission of address, however this is not required so it is suggested not to enable the DMA for transmission. The DMA request for reception is generated after completing the reception of data.
3. Slave mode as receiver - the DMA request for reception is generated after completing the reception of data (address or data).
4. Slave mode as transmitter - the DMA request for reception is generated after completing the reception of address, however this is not required so it is suggested not to enable the DMA for reception. The DMA request for transmission is generated after completing the transmission of data.

32.2.10 Error Interrupt Enable Register (I2Cn_IEIER)

The Error Interrupt Enable Register (I2Cn_IEIER) is used to enable AL and BER signals to generate an additional error interrupt output.

Error Interrupt Enable Register (I2Cn_IEIER)

Figure 32-15. Error Interrupt Enable Register (I2Cn_IEIER)

I2Cn_IEICR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	NSF[2]	NSF[1]	NSF[0]	NSFEN	EN	read0	read0	read0	read0	read0	ALCLR	BERCLR	INTCLR
Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32-13. Error Interrupt Enable Register (I2Cn_IEIER) bits

Bit Position	Bit Field Name	Bit Description
[15:13]	read0	-

Table 32-13. Error Interrupt Enable Register (I2Cn_IEIER) bits

Bit Position	Bit Field Name	Bit Description
[12:10]	NSF	Noise Filter Configuration bits These bits configure the maximum pulse width of single spikes on the SDA and SCL inputs that are suppressed by the built-in noise filter. '000': Suppress single spikes with a maximum pulse width of 0.5 peripheral clock cycles '001': Suppress single spikes with a maximum pulse width of 1 peripheral clock cycle '010': Suppress single spikes with a maximum pulse width of 1.5 peripheral clock cycles '011: Suppress single spikes with a maximum pulse width of 2 peripheral clock cycles '100': Suppress single spikes with a maximum pulse width of 2.5 peripheral clock cycles '101': Suppress single spikes with a maximum pulse width of 3 peripheral clock cycles '110': Suppress single spikes with a maximum pulse width of 3.5 peripheral clock cycles '111': Suppress single spikes with a maximum pulse width of 4 peripheral clock cycles The I2Cn_IEICR:NSF configuration is applicable when I2Cn_IEICR:NSFEN is set.
[9]	NSFEN	Noise Filter Enable bit '0': Noise filter disabled '1': Noise filter enabled This bit enables the noise filters for the SDA and SCL inputs. The noise filter will suppress single spikes according to the I2Cn_IEICR:NSF configuration. The NSFEN bit should be set to '1' if the interface is transmitting or receiving at data rates above 100 kbps.

Table 32-13. Error Interrupt Enable Register (I2Cn_IEIER) bits

Bit Position	Bit Field Name	Bit Description
[8]	EN	Enable bit This bit enables the I^2C interface operation. It can only be set by the user but may be cleared by the user and the hardware. '0': Interface disabled '1': Interface enabled When this bit is set to '0' all bits in the I2Cn_IBCSR register (except the I2Cn_IBCSR:BER and I2Cn_IBCSR:BEIE bits) are cleared. In addition, the module is disabled and the I^2C lines are left open. It is cleared by the hardware if a bus error occurs (i.e. I2Cn_IBCSR:BER = '1'). **Note:** The interface immediately stops transmitting or receiving if it is being disabled. This might leave the I^2C bus in an undesired state.
[7:3]	read0	-
[2]	ALCLR	Arbitration Lost Interrupt Flag Clear bit This bit clears the arbitration lost interrupt request flag. '0': No effect '1': Clears the I2Cn_IBCSR:AL register bit Reading this bit always returns '0'.
[1]	BERCLR	Bus Error Interrupt Flag Clear bit This bit clears the bus error interrupt request flag. '0': No effect '1': Clears the I2Cn_IBCSR:BER register bit Reading this bit always returns '0'.
[0]	INTCLR	Interrupt Request Flag Clear bit This bit clears the interrupt request flag. '0': No effect '1': Clears the I2Cn_IBCSR:INT and I2Cn_IBCSR:AL register bits Reading this bit always returns '0'.

32.3 Operation of I2C Interface

The I2C bus executes communication using two bi-directional bus lines, the serial data line (SDA), and serial clock line (SCL). The I2C Interface has two open-drain I/O pins (SDA/SCL) corresponding to these lines, enabling wired logic applications. The I2C Interface should not be used as a master in a multi-master system.

Start conditions

When the bus is free (i.e. I2Cn_IBCSR:BB = '0' and I2Cn_IBCSR:MSS = '0'), writing '1' to the I2Cn_IBCSR:MSS bit places the I2C Interface in master mode and generates a start condition.

If a '1' is written to the I2Cn_IBCSR:MSS bit while the bus is idle (i.e. I2Cn_IBCSR:MSS = '0' and I2Cn_IBCSR:BB = '0'), a start condition is generated and the contents of the I2Cn_IODAR:IODAR register (which should be address data) is sent.

Repeated start conditions can be generated by writing '1' to the I2Cn_IBCSR:SCC bit when in bus master mode and interrupt status is '1' (i.e. I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1').

If a '1' is written to the I2Cn_IBCSR:MSS bit while the bus is in use (i.e. I2Cn_IBCSR:BB = '1', I2Cn_IBCSR:TRX = '0', I2Cn_IBCSR:MSS = '0' and I2Cn_IBCSR:INT = '0'), the interface waits until the bus is free and then starts sending.

If the interface is addressed as a slave with write access (data reception) in the meantime, it will start sending after the transfer ended and the bus is free again. If the interface is sending data as a slave in the meantime, it will not start sending data if the bus is free again. It is important to check whether the interface was addressed as a slave (i.e. I2Cn_IBCSR:MSS = '0' and I2Cn_IBCSR:AAS = '1'), whether it sent the data byte successfully (i.e. I2Cn_IBCSR:MSS = '1') or failed to send the data byte (i.e. I2Cn_IBCSR:AL = '1') at the next interrupt.

Writing '1' to the I2Cn_IBCSR:MSS bit or I2Cn_IBCSR:SCC bit in any other situation has no significance.

Stop conditions

Writing '0' to the I2Cn_IBCSR:MSS bit in master mode (i.e. I2Cn_IBCSR:MSS = '1' and I2Cn_IBCSR:INT = '1') generates a stop condition and places the device in slave mode. Writing '0' to the I2Cn_IBCSR:MSS bit in any other situation has no significance.

After clearing the I2Cn_IBCSR:MSS bit, the interface tries to generate a stop condition which might fail if a certain external condition causes a signal transition from '1' to '0' at the SCL line before the generation of this stop condition. In this case, the I2Cn_IBCSR:AL bit is set to '1' and the interrupt is signalled at the end of the next byte.

Slave address detection

In slave mode after a start condition is generated the I2Cn_IBCSR:BB is set to '1' and data sent from the master device is received into the I2Cn_IODAR:IODAR register.

After the reception of eight bits, the contents of the I2Cn_IODAR:IODAR register is compared to the I2Cn_ISBMA:SA register using the bit mask stored in I2Cn_ISBMA:SM if the I2Cn_ISBMA:ENSB bit is '1'. If a match results, the I2Cn_IBCSR:AAS bit is set to '1' and an acknowledge signal is sent to the master. Then bit '0' of the received data (i.e. bit '0' of the I2Cn_IODAR:IODAR register) is stored in the I2Cn_IBCSR:TRX bit.

If the I2Cn_ITMK:ENTB bit is '1' and a 10-bit address header for write access (1 1 1 1 0 TA[9] TA[8] 0) is detected, the interface sends an acknowledge signal to the master and stores the last data bit in the I2Cn_IBCSR:TRX register. No interrupt is generated. The next transferred byte is then compared (using the bit mask stored in I2Cn_ITMK:TM) to the lower byte of the I2Cn_ITBA register. If a match is found, an acknowledge signal is sent to the master, the I2Cn_IBCSR:AAS bit is set and an interrupt is generated.

If the interface was addressed as slave and detects a repeated start condition, the I2Cn_IBCSR:AAS bit is set after reception of the 10-bit address header for read access (1 1 1 1 0 TA[9] TA[8] 1) and an interrupt is generated.

Since there are separate registers for the 10- and 7-bit address and their bit masks, it is possible to make the interface acknowledge on both addresses by setting the I2Cn_ISBMA:ENSB and I2Cn_ITMK:ENTB bits. The received slave address length (7- or 10-bit) may be determined by reading the I2Cn_ITMK:RAL bit (this bit is valid if the I2Cn_IBCSR:AAS bit is set only).

It is also possible to give the interface no slave address by setting both bits to '0' if it is only used as a master. All slave address bits may be masked with their corresponding Mask register (I2Cn_ITMK:TM or I2Cn_ISBMA:SM).

Slave address masking

Only the bits set to '1' in the Mask registers (I2Cn_ITMK:TM/I2Cn_ISBMA:SM) are used for address comparison, all other bits are ignored. The received slave address can be read from the I2Cn_ITBA (if ten bit address received, I2Cn_ITMK:RAL = '1') or I2Cn_ISBMA:SA (if seven bit address received, I2Cn_ITMK:RAL = '0') register if the I2Cn_IBCSR:AAS bit is '1'.

If the bit masks are cleared, the interface can be used as a bus monitor since it will always be addressed as slave.

Note: This is not a real bus monitor because it acknowledges upon any slave address reception, even if there is no other slave listening.

Addressing slaves

In master mode, after a start condition is generated the I2Cn_IBCSR:BB and I2Cn_IBCSR:TRX bits are set to '1' and the contents of the I2Cn_IODAR:IODAR register is sent in MSB first order. After address data is sent and an acknowledge signal was received from the slave device, bit '0' of the sent data (bit '0' of the I2Cn_IODAR:IODAR register after sending) is inverted, and stored in the I2Cn_IBCSR:TRX bit. Acknowledgement by the slave may be checked using the I2Cn_IBCSR:LRB bit. This procedure also applies to a repeated start condition.

In order to address a ten bit slave for write access, two bytes have to be sent. The first one is the ten bit address header which consists of the bit sequence (1 1 1 1 0 A9 A8 0), it is followed by the second byte containing the lower eight bits of the ten bit slave address (A7 to A0).

A 10-bit slave is accessed for reading by sending the above byte sequence and generating a repeated start condition (I2Cn_IBCSR:SCC bit) followed by a ten bit address header with read access (1 1 1 1 0 A9 A8 1).

Summary of the address data bytes:

7-bit slave, write access: Start condition - A6 A5 A4 A3 A2 A1 A0 0

7-bit slave, read access: Start condition - A6 A5 A4 A3 A2 A1 A0 1

10-bit slave, write access: Start condition - 1 1 1 1 0 A9 A8 0 - A7 A6 A5 A4 A3 A2 A1 A0

10-bit slave, read access: Start condition - 1 1 1 1 0 A9 A 8 0 - A7 A6 A5 A4 A3 A2 A1 A 0 - repeated start - 1 1 1 1 0 A9 A8 1

Acknowledgement

Acknowledge bits are sent from the receiver to the transmitter. The I2Cn_IBCSR:ACK bit can be used to select whether to send an acknowledgment when data bytes are received.

When data is sent in slave mode (read access from another master), if no acknowledgement is received from the master, the I2Cn_IBCSR:TRX bit is set to '0', and the device goes to receiving mode. This enables the master to generate a stop condition as soon as the slave has released the SCL line.

In master mode, acknowledgement by the slave can be checked by reading the I2Cn_IBCSR:LRB bit.

32.4 Programming flowcharts

The programming flowcharts for the 400 kHz I^{2}C Interface are shown below.

Programming flowcharts

Figure 32-16. Example of slave addressing and sending data

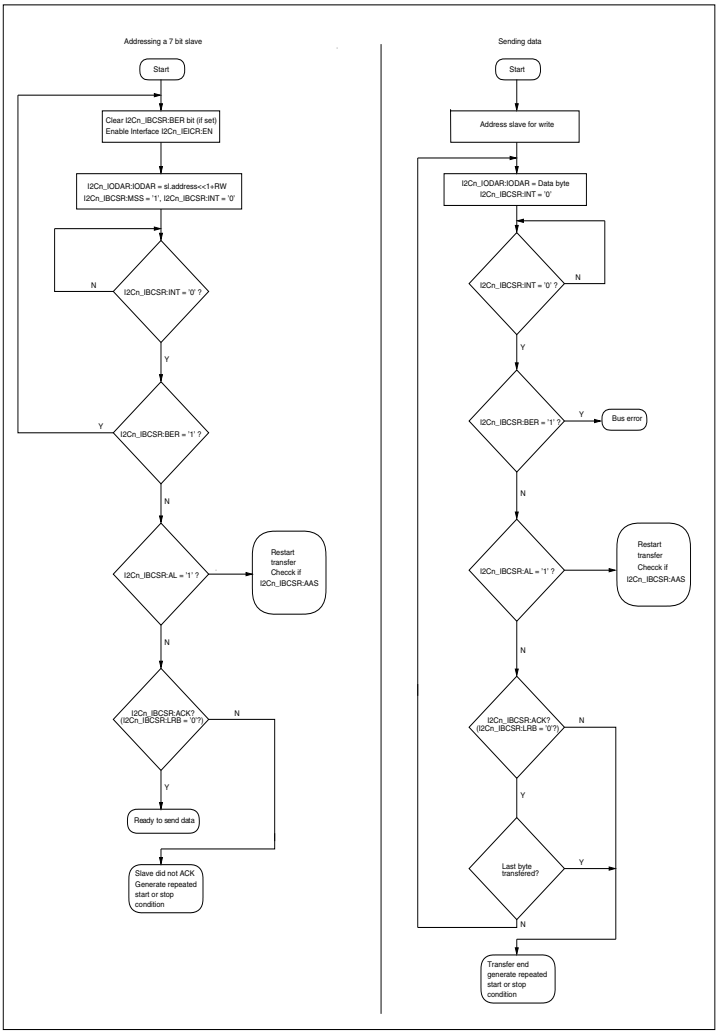
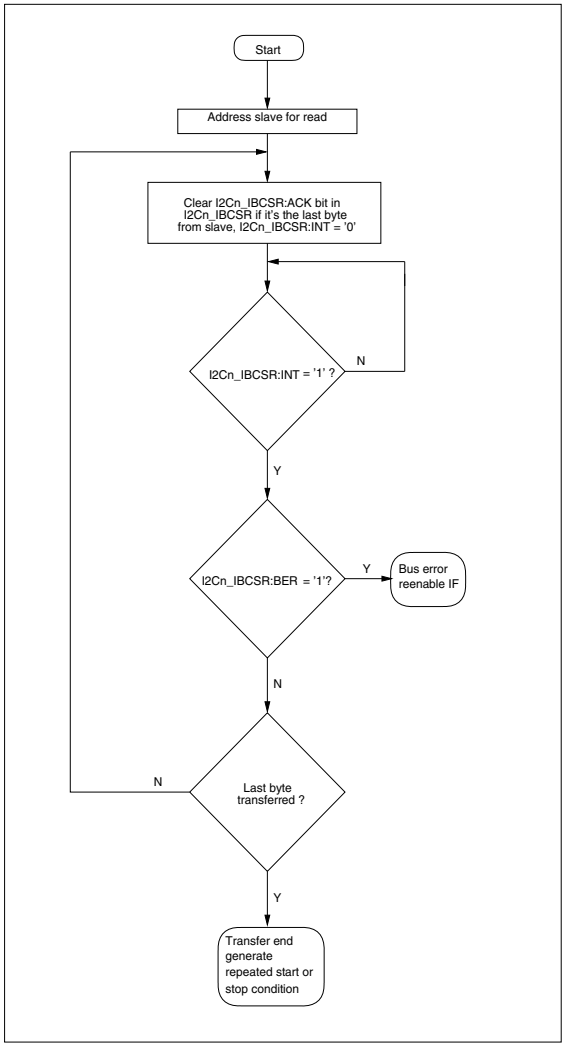


Figure 32-17. Example of receiving data



33. High Speed SPI Interface

This chapter explains the function and operation of the High Speed SPI Interface (HSSPI).

33.1 Outline of the HSSPI

This section describes the features and the block diagram of the HSSPI module.

Features of HSSPI

The HSSPI module provides various operating modes for interfacing to serial peripheral devices that use the de-facto standard SPI protocol. It also supports the new dual-bit and quad-bit SPI protocol. The features of HSSPI :

- It supports legacy, as well as, the dual-bit and quad-bit modes of SPI operation
- It supports up to four slave devices in master mode
- As well as the transfer rate, is the active level of slave select signal, polarity and phase of the serial clock per slave select
- External serial flash and serial SRAM devices can be memory-mapped to the address space of the MCU in 'command sequencer' mode 1
- In 'command sequencer' mode, memory accesses initiated by the MCU and other Advanced High-Performance Bus (AHB) masters are automatically converted to the serial memory read/write commands by the HSSPI
- 'Direct' mode allows the HSSPI to be used as a standard SPI through the First-in, First-out (FIFO) interface
- It supports the direct mode of operation

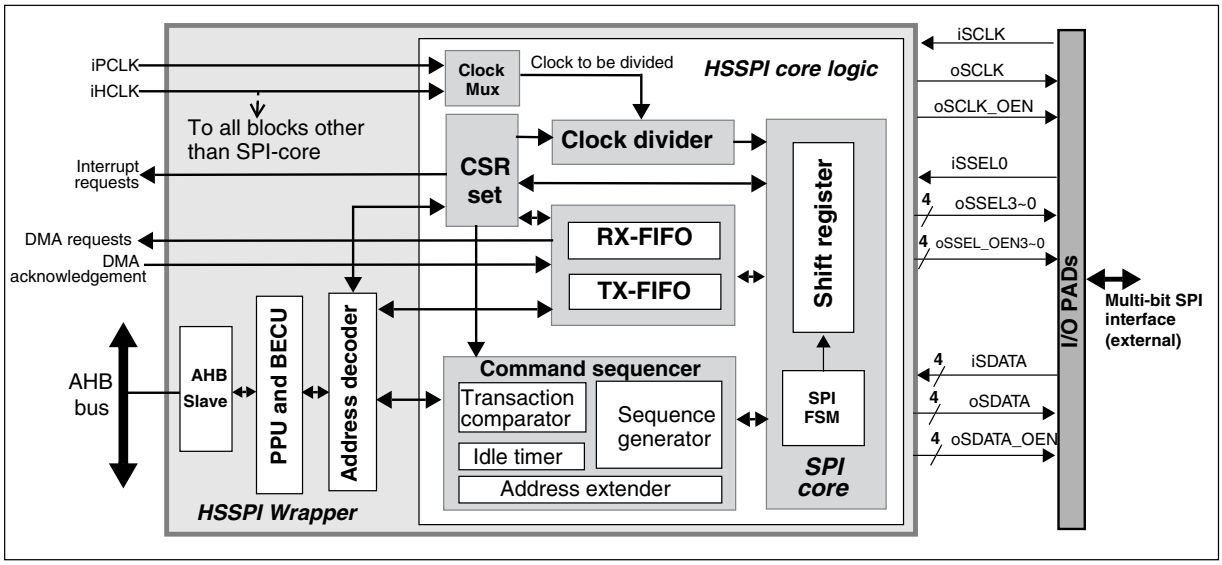
Note:

1: 'Command sequencer mode' may not be available in all devices. Check the device specific datasheet to see if your device supports the 'command sequencer mode'.

Block diagram of HSSPI

The Figure 33-1 shows the internal block diagram of the HSSPI module.

Figure 33-1. Block diagram of HSSPI



- HSSPI register set

  The operation of the HSSPI can be controlled and monitored through its Configuration and Status Register (CSR) set. For details of the CSR refer to 33.6 Address map of HSSPI.

- SPI core and clock divider

  The SPI core is the protocol engine, that drives/samples the serial interface. The SPI communication-related attributes (serial clock frequency, serial clock phase, polarity, etc.) are configured in the CSR. On the host side, based on whether the HSSPI is operating in direct mode or in command sequencer mode, the SPI core connects either with the FIFOs or with the command sequencer. When operating as an SPI master, HSSPI can initiate serial transfers with upto four SPI slaves, which are connected to the four slave-select lines: SSEL0, SSEL1, SSEL2, and SSEL3. When operating as an SPI slave, HSSPI can respond to serial transfers initiated by the external SPI master when its SSEL0 pin is asserted. The internal clock divider is used to derive the serial clock output (SCLK), which is used when the HSSPI acts as an SPI master. The 'clock mux' multiplexes between the two source clocks, AHB clock (i.e. iHCLK) and the peripheral clock (i.e. iPCLK),, and selects one of them as an input to the clock divider depending on the configuration in CSR.

- AHB interface and address decoding

  AHB masters can access the HSSPI module through its AHB slave interface. The address decoder decodes the AHB address bus. If the access is for a register, the address is routed to the CSRs. If the AHB access is for a serial memory device in command sequencer mode, which is mapped onto one of the four slave-select lines, the memory address is passed on to the command sequencer.

- TX-FIFO and RX-FIFO

  Internally the HSSPI has two FIFOs for temporary storage, one for the data to be transmitted and one for the data to be received. Each FIFO is 16 locations deep and has a data width of 32 bits. The FIFOs are used by the HSSPI in the direct mode of operation.

- Command sequencer

  The command sequencer maps the external serial memory devices onto the address space of MCU. It consists of: a transaction comparator; an address extender; an idle timer and sequence generator.
  Whenever there is an access to the memory-mapped serial memory device, the sequence generator initiates a serial transaction for accessing the external memory device. The address for the serial memory device is generated by the 'address extender' logic block. This concatenates some of the bits from the AHB address bus with the HSSPI address extension register. When a memory read/write transaction on the serial interface is complete, the sequence generator responds to the AHB master, cuts the serial clock but keeps asserting the corresponding slave select output for a period defined by the idle timer.
  The sequence generator switches on the serial clock to access the next location of the serial memory during the idle time-out period if the address and the access type (R/W) match the previous transaction in the transaction comparator.

- Peripheral protection and error collection

  The peripheral protection logic in the HSSPI protects the HSSPI CSRs from illegal AHB accesses, based on the access levels defined in the PPU module of the MCU. Faults are signalled by HSSPI by generating a bus error response, so that the transaction initiator can know the outcome of the transfer.

33.2 HSSPI registers

This section describes the registers of the HSSPI in detail.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of HSSPI

The following registers are available for each instance of the HSSPI:

- HSSPI Module Control Register (HSSPIn_MCTRL)
- HSSPI Peripheral Communication Configuration Register 0~3 (HSSPIn_PCC0~3)
- HSSPI TX Interrupt Flag Register (HSSPIn_TXF)
- HSSPI TX Interrupt Enable Register (HSSPIn_TXE)
- HSSPI TX Interrupt Clear Register (HSSPIn_TXC)
- HSSPI RX Interrupt Flag Register (HSSPIn_RXF)
- HSSPI RX Interrupt Enable Register (HSSPIn_RXE)
- HSSPI RX Interrupt Clear Register (HSSPIn_RXC)
- HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF)
- HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC)
- HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG)
- HSSPI Direct Mode DMA Enable Register (HSSPIn_DMDMAEN)
- HSSPI Direct Mode Start Register (HSSPIn_DMSTART)
- HSSPI Direct Mode Stop Register (HSSPIn_DMSTOP)
- HSSPI Direct Mode Peripheral Select Register (HSSPIn_DMPSEL)
- HSSPI Direct Mode Transfer Protocol Register (HSSPIn_DMTRP)
- HSSPI Direct Mode Byte Count Control Register (HSSPIn_DMBCC)
- HSSPI Direct Mode Byte Count Status Register (HSSPIn_DMBCS)
- HSSPI Direct Mode Status Register (HSSPIn_DMSTATUS)
- HSSPI Transmit Bit Count Register (HSSPIn_TXBITCNT)
- HSSPI Receive Bit Count Register (HSSPIn_RXBITCNT)
- HSSPI RX Shift Register (HSSPIn_RXSHIFT)
- HSSPI TX-FIFO Registers (HSSPIn_TXFIFO0~15)
- HSSPI RX-FIFO Registers (HSSPIn_RXFIFO0~15)
- HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG)
- HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG)
- HSSPI Command Sequencer Idle Time Register (HSSPIn_CSITIME)
- HSSPI Command Sequencer Address Extension Register (HSSPIn_CSAEXT)
- HSSPI Read Command Sequence Data/Control Register 0~7 (HSSPIn_RDCSDC0~7)
- HSSPI Write Command Sequence Data/Control Register 0~7 (HSSPIn_WRCSDC0~7)
- HSSPI Module ID Register (HSSPIn_MID)

Memory layout of HSSPI registers

Table 33-1. Memory layout of HSSPI registers

Offset	+3	+2	+1	+0
0x00000000	HSSPIn_MCTRL 00000000 00000000 00000000 00000000			

Table 33-1. Memory layout of HSSPI registers

Offset	+3	+2	+1	+0
0x00000004	HSSPIn_PCC0 00000000 00000001 00000000 00000000			
0x00000008	HSSPIn_PCC1 00000000 00000001 00000000 00000000			
0x0000000C	HSSPIn_PCC2 00000000 00000001 00000000 00000000			
0x00000010	HSSPIn_PCC3 0 00000000 00000011 00000000 00000000			
0x00000014	HSSPIn_TXF 00000000 00000000 00000000 00000000			
0x00000018	HSSPIn_TXE 00000000 00000000 00000000 00000000			
0x0000001C	HSSPIn_TXC 00000000 00000000 00000000 00000000			
0x00000020	HSSPIn_RXF 00000000 00000000 00000000 00000000			
0x00000024	HSSPIn_RXE 00000000 00000000 00000000 00000000			
0x00000028	HSSPIn_RXC 00000000 00000000 00000000 00000000			
0x0000002C	HSSPIn_FAULTF 00000000 00000000 00000000 00000000			
0x00000030	HSSPIn_FAULTC 00000000 00000000 00000000 00000000			
0x00000034	read0 00000000 00000000		HSSPIn_DMDMAEN 00000000	HSSPIn_DMCFG 00000001
0x00000038	HSSPIn_DMTRP 00000000	HSSPIn_DMPSEL 00000000	HSSPIn_DMSTOP 00000000	HSSPIn_DMSTART 00000000
0x0000003C	HSSPIn_DMBCS 00000000 00000000		HSSPIn_DMBCC 00000000 00000000	
0x00000040	HSSPIn_DMSTATUS 00000000 00000000 00000000 00000000			
0x00000044	read0 00000000 00000000		HSSPIn_RXBITCNT 00000000	HSSPIn_TXBITCNT 00000000
0x00000048	HSSPIn_RXSHIFT 00000000 00000000 00000000 00000000			
0x0000004C	HSSPIn_FIFOCFG 00000000 00000000 00000000 01110111			

Table 33-1. Memory layout of HSSPI registers

Offset	+3	+2	+1	+0
0x00000050	HSSPIn_TXFIFO0 00000000 00000000 00000000 00000000			
0x00000054	HSSPIn_TXFIFO1 00000000 00000000 00000000 00000000			
0x00000058	HSSPIn_TXFIFO2 00000000 00000000 00000000 00000000			
0x0000005C	HSSPIn_TXFIFO3 00000000 00000000 00000000 00000000			
0x00000060	HSSPIn_TXFIFO4 00000000 00000000 00000000 00000000			
0x00000064	HSSPIn_TXFIFO5 00000000 00000000 00000000 00000000			
0x00000068	HSSPIn_TXFIFO6 00000000 00000000 00000000 00000000			
0x0000006C	HSSPIn_TXFIFO7 00000000 00000000 00000000 00000000			
0x00000070	HSSPIn_TXFIFO8 00000000 00000000 00000000 00000000			
0x00000074	HSSPIn_TXFIFO9 00000000 00000000 00000000 00000000			
0x00000078	HSSPIn_TXFIFO10 00000000 00000000 00000000 00000000			
0x0000007C	HSSPIn_TXFIFO11 00000000 00000000 00000000 00000000			
0x00000080	HSSPIn_TXFIFO12 00000000 00000000 00000000 00000000			
0x00000084	HSSPIn_TXFIFO13 00000000 00000000 00000000 00000000			
0x00000088	HSSPIn_TXFIFO14 00000000 00000000 00000000 00000000			
0x0000008C	HSSPIn_TXFIFO15 00000000 00000000 00000000 00000000			
0x00000090	HSSPIn_RXFIFO0 00000000 00000000 00000000 00000000			
0x00000094	HSSPIn_RXFIFO1 00000000 00000000 00000000 00000000			
0x00000098	HSSPIn_RXFIFO2 00000000 00000000 00000000 00000000			

Table 33-1. Memory layout of HSSPI registers

Offset	+3	+2	+1	+0
0x0000009C	HSSPIn_RXFIFO3 00000000 00000000 00000000 00000000			
0x000000A0	HSSPIn_RXFIFO4 00000000 00000000 00000000 00000000			
0x000000A4	HSSPIn_RXFIFO5 00000000 00000000 00000000 00000000			
0x000000A8	HSSPIn_RXFIFO6 00000000 00000000 00000000 00000000			
0x000000AC	HSSPIn_RXFIFO7 00000000 00000000 00000000 00000000			
0x000000B0	HSSPIn_RXFIFO8 00000000 00000000 00000000 00000000			
0x000000B4	HSSPIn_RXFIFO9 00000000 00000000 00000000 00000000			
0x000000B8	HSSPIn_RXFIFO10 00000000 00000000 00000000 00000000			
0x000000BC	HSSPIn_RXFIFO11 00000000 00000000 00000000 00000000			
0x000000C0	HSSPIn_RXFIFO12 00000000 00000000 00000000 00000000			
0x000000C4	HSSPIn_RXFIFO13 00000000 00000000 00000000 00000000			
0x000000C8	HSSPIn_RXFIFO14 00000000 00000000 00000000 00000000			
0x000000CC	HSSPIn_RXFIFO15 00000000 00000000 00000000 00000000			
0x000000D0	HSSPIn_CSCFG 00000000 00000000 00000000 00000000			
0x000000D4	HSSPIn_CSITIME 00000000 00000000 11111111 11111111			
0x000000D8	HSSPIn_CSAEXT 00000000 00000000 00000000 00000000			
0x000000DC	HSSPIn_RDCSDC1 00000000 00000000		HSSPIn_RDCSDC0 00000000 00000000	
0x000000E0	HSSPIn_RDCSDC3 00000000 00000000		HSSPIn_RDCSDC2 00000000 00000000	
0x000000E4	HSSPIn_RDCSDC5 00000000 00000000		HSSPIn_RDCSDC4 00000000 00000000	

Table 33-1. Memory layout of HSSPI registers

Offset	+3	+2	+1	+0
0x000000E8	HSSPIn_RDCSDC7 00000000 00000000		HSSPIn_RDCSDC6 00000000 00000000	
0x000000EC	HSSPIn_WRCSDC1 00000000 00000000		HSSPIn_WRCSDC0 00000000 00000000	
0x000000F0	HSSPIn_WRCSDC3 00000000 00000000		HSSPIn_WRCSDC2 00000000 00000000	
0x000000F4	HSSPIn_WRCSDC5 00000000 00000000		HSSPIn_WRCSDC4 00000000 00000000	
0x000000F8	HSSPIn_WRCSDC7 00000000 00000000		HSSPIn_WRCSDC6 00000000 00000000	
0x000000FC	HSSPIn_MID 00000000 00000000 00000000 00000001			

33.2.1 HSSPI Module Control Register (HSSPIn_MCTRL)

The HSSPI Module Control Register controls the HSSPI module. It contains essential bits like the module enable bit, the command sequencer enable bit and the debug enable bit. The software can enable/disable the module operation by using this register.

HSSPI Module Control Register (HSSPIn_MCTRL)

Figure 33-2. HSSPI Module Control Register (HSSPI_MTCRL)

HSSPIn_MCTRL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	MES	CDSS	DBGEN	CSEN	MEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-2. HSSPI Module Control Register (HSSPIn_MTCRL) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4]	MES	Module Enable Status '0': Module is completely disabled and it has entered the power saving mode '1': Module is enabled
[3]	CDSS	Clock Division Source Select When HS_SPI is in master mode, the internal clock divider can divide either the AHB clock (i.e. iHCLK) or the peripheral clock (i.e. iPCLK). The CDSS bit decides which clock is divided by the clock divider. This field is not used in slave mode. '0': Clock divider divides the iHCLK '1': Clock divider divides the iPCLK

Table 33-2. HSSPI Module Control Register (HSSPIn_MTCRL) bits

Bit Position	Bit Field Name	Bit Description
[2]	DBGEN	Debug Enable This bit is used to enable/disable debug mode for HSSPI. '0': Debug mode disabled '1': Debug mode enabled This bit takes effect only in the direct mode of operation (i.e. HSSPIn_MCTRL:CSEN = '0'), when HSSPI is configured as a master (i.e. HSSPIn_DMCFG:MST = '1'). When DBGEN is set to '1' and the processor is in debug state and the HSSPI is working as a master in the direct mode of operation, the serial interface is halted by stopping the SCLK output. For the definition of the debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.
[1]	CSEN	Command Sequencer Enable '0': Direct mode is enabled. Command sequencer is disabled '1': Command sequencer is enabled. Direct mode is disabled **Note:** To check whether the command sequencer mode is available in a particular device, refer to the device specific datasheet. If command sequencer is not available in the device, the CSEN bit is read-only and its value is '0'.
[0]	MEN	Module Enable '0': Module is disabled. The HSSPI module enters power saving mode and the interrupt registers will not update. All serial I/O signals are tri-stated by the HSSPI '1': Module is enabled After configuring the HSSPI, the software must set this bit to '1' to enable the HSSPI in operating mode. When the software resets this bit: a. In direct mode - As a master the HSSPI stops further SPI transfers after the slave select is released (if it is already asserted). As a slave, HSSPI does not respond to any SPI transfers after the slave select is released (if it is already asserted). After the slave select is released, it internally enters a power saving mode, by gating the iHCLK and the iPCLK clocks to some of its internal logic blocks. b. In command sequencer mode - The HSSPI generates an unmapped memory access fault interrupt if any further AHB access to memory mapped devices are received. It does not initiate any commands on the serial interface. After the slave select has been released, it internally enters power saving mode by gating the iHCLK and the iPCLK clocks to some of its internal logic blocks.

33.2.2 HSSPI Peripheral Communication Configuration Register 0~3 (HSSPIn_PCC0~3)

The HSSPI Peripheral Communication Configuration Registers 0~3 control the attributes related to serial communication on slave select 0~3. The software must initialize these registers with attributes that match the communication attributes of the serial peripheral that is to be interfaced to the corresponding slave select line (0 ~ 3) of the HSSPI. In master mode, each of the four registers is used. In slave mode, only the HSSPIn_PCC0 register is used by the HSSPI. Only the HSSPIn_PCC0 register is described here. Other registers (i.e. HSSPIn_PCC1, HSSPIn_PCC2, and HSSPIn_PCC3) have identical bit fields.

HSSPI Peripheral Communication Configuration Register 0 (HSSPIn_PCC0)

Figure 33-3. HSSPI Peripheral Communication Configuration Register 0 (HSSPIn_PCC0)

HSSPIn_PCC0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SAFESYNC	CDRS[6]	CDRS[5]	CDRS[4]	CDRS[3]	CDRS[2]	CDRS[1]	CDRS[0]	read0	SDIR	SS2CD[1]	SS2CD[0]	SSPOL	RTM	ACES	CPOL	CPHA
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-3. HSSPI Peripheral Communication Configuration Register 0 (HSSPIn_PCC0) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	SAFESYNC	Safe Synchronisation for Peripheral 0 This bit is valid only when the HSSPI is configured as an SPI master in direct mode or command sequencer mode. '0': The module operates normally. Pre-determined delay for safe synchronisation of data is not added by HSSPI during the serial transfers '1': The module implements the safe synchronisation of data while serial communication with peripheral '0' is taking place

Table 33-3. HSSPI Peripheral Communication Configuration Register 0 (HSSPIn_PCC0) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	CDRS	Clock Division Ratio Select of Peripheral 0 When the HSSPI is configured as an SPI master in direct mode or in command sequencer mode, this field decides the clock division ratio of the internal clock divider. This field is not used when the HSSPI is configured in SPI slave mode. 0: Clock divider is bypassed. SCLK is same as source clock, selected by HSSPIn_MCTRL:CDSS bit 1: Divide by 2 2: Divide by 4 3: Divide by 6 ... 127: Divide by 254 In general, for a non-zero value of CDRS, the source clock frequency (i.e. Fi) is divided by twice the CDRS value, to get the derived clock frequency (i.e. Fo). Fo = Fi/(2 x CDRS) The value of the CDRS bit should be chosen such that the resultant serial clock frequency is not more than the frequency of the AHB clock.
[8]	read0	-
[7]	SDIR	Shift Direction of Peripheral 0 The SDIR bit decides the bit transmission order within a field. It bit does not affect the position of the most significant bit and least significant bit in the data registers. Read or write access to data registers always has the least significant bit in bit'0'. '0': Most significant bit is transmitted first '1': Least significant bit is transmitted first

Table 33-3. HSSPI Peripheral Communication Configuration Register 0 (HSSPIn_PCC0) bits

Bit Position	Bit Field Name	Bit Description
[6:5]	SS2CD	Slave-Select to Clock Delay of Peripheral 0 This bit is used only when the HSSPI is configured as an SPI master in direct mode or in command sequencer mode. It defines a setup time for the slave device. By delaying the toggling of SCLK, the HSSPI delays the data transmission (of the slave) from the chip select active edge by a multiple of SCLK cycles. If HSSPIn_PCC0~3:CPHA = '0', the delay between assertion of slave select and the first edge on the SCLK is given by: (SS2CD + 1 + 0.5) number of clock periods of SCLK. If HSSPIn_PCC0~3:CPHA = '1', the delay between assertion of slave select and the first edge on the SCLK is given by: (SS2CD + 1) number of clock periods of SCLK. These formulas change in devices that implement a different HSSPI revision (3 or 4): If HSSPIn_PCC0~3:CPHA = '0', the delay between assertion of slave select and the first edge on the SCLK is given by: (SS2CD + 3 +0.5) number of clock periods of SCLK If HSSPIn_PCC0~3:CPHA = '1', the delay between assertion of slave select and the first edge on the SCLK is given by: (SS2CD + 3) number of clock periods of SCLK. When the slave select becomes active, it has to prepare the data transfer within the delay time defined by the SS2CD bits.
[4]	SSPOL	Slave Select Polarity of Peripheral 0 This bit is used to decide the polarity of the slave select (i.e. SSEL0) signal. '0': SSEL0 is held high during the default state. The signal is active low '1': SSEL0 is held low during the default state. The signal is active high This bit must not be changed while the module is enabled (i.e. HSSPIn_MCTRL:MES='1').
[3]	RTM	Use Retimed Clock for Capturing the Data from Peripheral 0 This bit must be set to '1' if the serial device interfaced with the HSSPI provides tight setup or hold margins to the HSSPI. This bit takes effect only when the HSSPI is configured as an SPI master in direct mode or command sequencer mode. '0': Do not use the retimed clock to capture the serial data '1': Use the retimed clock to capture the serial data The HSSPIn_PCC0:CPHA, HSSPIn_PCC0:CPOL, HSSPIn_PCC0:ACES and RTM bits together decide the clocking mode of the HSSPI serial interface.

Table 33-3. HSSPI Peripheral Communication Configuration Register 0 (HSSPIn_PCC0) bits

Bit Position	Bit Field Name	Bit Description
[2]	ACES	Active Clock Edges are Same on Peripheral 0 This bit decides whether the active edges of the clock used for launching and capturing data are same or opposite. This bit takes effect only when the HSSPI is configured as an SPI master in direct mode or in command sequencer mode. '0': Launching and capturing of data are done on alternate (i.e. opposite) clock edges '1': Launching and capturing of data are done on the same clock edges The HSSPIn_PCC0:CPHA, HSSPIn_PCC0:CPOL, ACES, and HSSPIn_PCC0:RTM bits together decide the clocking mode of HS_SPI serial interface. Do not set ACES to '1' in slave mode (HSSPIn_D-MCFG:MST='0'). Do not set ACES to '1' if HSSPIn_DMTRP:TRP is '0000'.
[1]	CPOL	Clock Polarity of Peripheral 0 '0': SCLK is held low during its default state '1': SCLK is held high during its default state The HSSPIn_PCC0:CPHA, CPOL, HSSPIn_PCC0:ACES, and HSSPIn_PCC0:RTM bits together decide the clocking mode of the HSSPI serial interface.
[0]	CPHA	Clock Phase of Peripheral 0 '0': Input data are sampled on odd numbered edges of the serial clock '1': Input data are sampled on even numbered edges of the serial clock The CPHA, HSSPIn_PCC0:CPOL, HSSPIn_PCC0:ACES and HSSPIn_PCC0:RTM bits together decide the clocking mode of the HSSPI serial interface.

33.2.3 HSSPI TX Interrupt Flag Register (HSSPIn_TXF)

The HSSPI TX Interrupt Flag Register indicates the status of the TX interrupt flags. These interrupt flags are set in the direct mode of operation only. Software can enable these interrupts and wait for their assertion or it can use them in polling mode.

HSSPI TX Interrupt Flag Register (HSSPIn_TXF)

Figure 33-4. HSSPI TX Interrupt Flag Register (HSSPIn_TXF)

HSSPIn_TXF																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TSSRS	TFMTS	TFLETS	TFUS	TFOS	TFES	TFFS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-4. HSSPI TX Interrupt Flag Register (HSSPIn_TXF) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	TSSRS	Slave Select Released This interrupt flag indicates that the slave select line has been released by the SPI master. This interrupt flag triggers the TX interrupt signal, if it is enabled in HSSPIn_TXE:TSSRE.
[5]	TFMTS	TX-FIFO Fill Level is More Than Threshold This interrupt flag is set with every AHB clock if the TX-FIFO fill level is more than the configured TX-FIFO threshold value. i.e. HSSPIn_DMSTATUS:TXFLEVEL is greater than HSSPIn_FIFO-CFG:TXFTH. This interrupt flag triggers the TX interrupt signal if it is enabled in HSSPIn_TXE:TFMTE.
[4]	TFLETS	TX-FIFO Fill Level is Less Than or Equal to Threshold This interrupt flag is set with every AHB clock if the TX-FIFO fill level is less than or equal to the configured TX-FIFO threshold value. i.e. HSSPIn_DMSTATUS:TXFLEVEL is less than or equal to HSS-PIn_FIFOCFG:TXFTH. This interrupt flag triggers the TX interrupt signal if it is enabled in HSSPIn_TXE:TFLETE.

Table 33-4. HSSPI TX Interrupt Flag Register (HSSPIn_TXF) bits

Bit Position	Bit Field Name	Bit Description
[3]	TFUS	TX-FIFO Underrun This interrupt flag indicates that the TX-FIFO is underrun. The TX-FIFO underrun condition happens when TX-FIFO is read by the SPI core while empty. This condition may happen during the slave mode of operation. This interrupt flag triggers the TX interrupt signal if it is enabled in HSSPIn_TXE:TFUE.
[2]	TFOS	TX-FIFO Overrun This interrupt flag indicates that the TX-FIFO is overrun. The TX-FIFO overrun condition happens when HSSPIn_TX-FIFO0~15 register is written to by the software while the TX-FIFO is full. This interrupt flag triggers the TX interrupt signal if it is enabled in HSSPIn_TXE:TFOE.
[1]	TFES	TX-FIFO and Shift Register is Empty This interrupt flag is set with every AHB clock, if the TX-FIFO and the TX shift register (in the SPI core) are empty. This interrupt flag triggers the TX interrupt signal if it is enabled in HSSPIn_TXE:TFEE. **Note:** As illustrated in Figure 33-56 and described in the subsequent text, in slave mode the TX-FIFO should never be empty after a transfer has been completed. This then means that the TFES flag should never be set. If it is set, then there is a problem with software operation in slave mode.
[0]	TFFS	TX-FIFO Full This interrupt flag is set with every AHB clock if the TX-FIFO is full. This interrupt flag triggers the TX interrupt signal if it is enabled in HSSPIn_TXE:TFFE.

33.2.4 HSSPI TX Interrupt Enable Register (HSSPIn_TXE)

The HSSPI TX Interrupt Enable Register decides whether or not the interrupt flags in HSSPIn_TXF register trigger the TX interrupt.

The software must enable these flags to wait for the assertion of the TX interrupt signal.

HSSPI TX Interrupt Enable Register (HSSPIn_TXE)

Figure 33-5. HSSPI TX Interrupt Enable Register (HSSPIn_TXE)

HSSPIn_TXE																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TSSRE	TFMTE	TFLETE	TFUE	TFOE	TFEE	TFFE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-5. HSSPI TX Interrupt Enable Register (HSSPIn_TXE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	TSSRE	Slave Select Released Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TSSRS interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TSSRS interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TSSRS interrupt flag triggers the TX interrupt signal
[5]	TFMTE	TX-FIFO Fill Level is More Than Threshold Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TFMTS interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TFMTS interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TFMTS interrupt flag triggers the TX interrupt signal

Table 33-5. HSSPI TX Interrupt Enable Register (HSSPIn_TXE) bits

Bit Position	Bit Field Name	Bit Description
[4]	TFLETE	TX-FIFO Fill Level is Less Than or Equal To Threshold Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TFLETS interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TFLETS interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TFLETS interrupt flag triggers the TX interrupt signal
[3]	TFUE	TX-FIFO Underrun Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TFUS interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TFUS interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TFUS interrupt flag triggers the TX interrupt signal
[2]	TFOE	TX-FIFO Overrun Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TFOS interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TFOS interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TFOS interrupt flag triggers the TX interrupt signal
[1]	TFEE	TX-FIFO Empty Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TFES interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TFES interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TFES interrupt flag triggers the TX interrupt signal
[0]	TFFE	TX-FIFO Full Interrupt Enable This bit decides whether or not the HSSPIn_TXF:TFFS interrupt flag is routed on the TX interrupt signal. '0': The HSSPIn_TXF:TFFS interrupt flag does not trigger the TX interrupt signal '1': The HSSPIn_TXF:TFFS interrupt flag triggers the TX interrupt signal

33.2.5 HSSPI TX Interrupt Clear Register (HSSPIn_TXC)

The HSSPI TX Interrupt Clear Register is used to clear the interrupt flags set in the HSSPIn_TXF register. By writing '1' to a bit in this register, the software can clear the corresponding flag in the HSSPIn_TXF register.

HSSPI TX Interrupt Clear Register (HSSPIn_TXC)

Figure 33-6. HSSPI TX Interrupt Clear Register (HSSPIn_TXC)

HSSPIn_TXC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TSSRC	TFMTC	TFLETC	TFUC	TFOC	TFEC	TFFC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-6. HSSPI TX Interrupt Clear Register (HSSPIn_TXC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	TSSRC	Slave Select Released Interrupt Clear This bit is used to clear the HSSPIn_TXF:TSSRS interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TSSRS interrupt flag Read returns '0'.
[5]	TFMTC	TX-FIFO Fill Level More Than Threshold Interrupt Clear This bit is used to clear the HSSPIn_TXF:TFMTS interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TFMTS interrupt flag Read returns '0'.
[4]	TFLETC	TX-FIFO Fill Level Less Than or Equal to Threshold Interrupt Clear This bit is used to clear the HSSPIn_TXF:TFLETS interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TFLETS interrupt flag Read returns '0'.

Table 33-6. HSSPI TX Interrupt Clear Register (HSSPIn_TXC) bits

Bit Position	Bit Field Name	Bit Description
[3]	TFUC	TX-FIFO Underrun Interrupt Clear This bit is used to clear the HSSPIn_TXF:TFUS interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TFUS interrupt flag Read returns '0'.
[2]	TFOC	TX-FIFO Overrun Interrupt Clear This bit is used to clear the HSSPIn_TXF:TFOS interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TFOS interrupt flag Read returns '0'.
[1]	TFEC	TX-FIFO Empty Interrupt Clear This bit is used to clear the HSSPIn_TXF:TFES interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TFES interrupt flag Read returns '0'.
[0]	TFFC	TX-FIFO Full Interrupt Clear This bit is used to clear the HSSPIn_TXF:TFFS interrupt flag. '0': No effect '1': Clears the HSSPIn_TXF:TFFS interrupt flag Read returns '0'.

33.2.6 HSSPI RX Interrupt Flag Register (HSSPIn_RXF)

The HSSPI RX Interrupt Flag register indicates the status of the RX interrupt flags. These interrupt flags are set in the direct mode of operation only. Software can enable these interrupts and wait for their assertion or it can use them in polling mode.

HSSPI RX Interrupt Flag Register (HSSPIn_RXF)

Figure 33-7. HSSPI RX Interrupt Flag Register (HSSPIn_RXF)

HSSPIn_RXF																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RSSRS	RFMTS	RFLETS	RFUS	RFOS	RFES	RFFS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-7. HSSPI RX Interrupt Flag Register (HSSPIn_RXF) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	RSSRS	Slave Select Released This interrupt flag indicates that the slave select line is released by the SPI master. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RSSRE.
[5]	RFMTS	RX-FIFO Fill Level is More Than Threshold This interrupt flag is set with every AHB clock if the RX-FIFO fill level is more than the configured RX-FIFO threshold value. i.e. HSSPIn_DMSTATUS:RXFLEVEL is greater than HSSPIn_FIFOCFG:RXFT. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RFMTE.
[4]	RFLETS	RX-FIFO Fill Level is Less Than or Equal to Threshold This interrupt flag is set with every AHB clock if the RX-FIFO fill level is less than or equal to the configured RX-FIFO threshold value. i.e. HSSPIn_DMSTATUS:RXFLEVEL is less than or equal to HSSPIn_FIFOCFG:RXFTH. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RFLETE.

Table 33-7. HSSPI RX Interrupt Flag Register (HSSPIn_RXF) bits

Bit Position	Bit Field Name	Bit Description
[3]	RFUS	RX-FIFO Underrun This interrupt flag indicates that the RX-FIFO is under-run. The RX-FIFO underrun condition happens when HSSPIn_RXFIFO0~15 register is read (by an AHB master other than the DAP controller) while the RX-FIFO is empty. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RFUE. **Note:** This flag is not set when the DAP controller reads the HSSPIn_RXFIFO0~15 register while the RX-FIFO is empty.
[2]	RFOS	RX-FIFO Overrun This interrupt flag indicates that the RX-FIFO is overrun. The RX-FIFO overrun condition happens when RX-FIFO is written to by the SPI core while full. This condition may happen during the slave mode of operation. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RFOE.
[1]	RFES	RX-FIFO Empty This interrupt flag is set with every AHB clock if the RX-FIFO is empty. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RFEE.
[0]	RFFS	RX-FIFO Full This interrupt flag is set with every AHB clock if the RX-FIFO is full. This interrupt flag triggers the RX interrupt signal if it is enabled in HSSPIn_RXE:RFFE.

33.2.7 HSSPI RX Interrupt Enable Register (HSSPIn_RXE)

The HSSPI RX Interrupt Enable Register decides whether or not the interrupt flags in HSSPIn_TXF register trigger the RX interrupt. The software must enable these flags, if it wants to wait for the assertion of the RX interrupt signal.

HSSPI RX Interrupt Enable Register (HSSPIn_RXE)

Figure 33-8. HSSPI RX Interrupt Enable Register (HSSPIn_RXE)

HSSPIn_RXE																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RSSRE	RFMTE	RFLETE	RFUE	RFOE	RFEE	RFFE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-8. HSSPI RX Interrupt Enable Register (HSSPIn_RXE) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	RSSRE	Slave Select Released Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RSSRS interrupt flag is routed on the RX interrupt signal. '0': The HSSPIn_RXF:RSSRS interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RSSRS interrupt flag triggers the RX interrupt signal
[5]	RFMTE	RX-FIFO Fill Level is More Than Threshold Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RFMTS interrupt flag is routed on the RX interrupt signal. '0': The HSSPIn_RXF:RFMTS interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RFMTS interrupt flag triggers the RX interrupt signal

Table 33-8. HSSPI RX Interrupt Enable Register (HSSPIn_RXE) bits

Bit Position	Bit Field Name	Bit Description
[4]	RFLETE	RX-FIFO Fill Level is Less Than or Equal To Threshold Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RFLETS interrupt flag is routed on the RX interrupt signal. '0': The HSSPIn_RXF:RFLETS interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RFLETS interrupt flag triggers the RX interrupt signal
[3]	RFUE	RX-FIFO Underrun Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RFUS interrupt flag is routed on the RX interrupt signal. '0': The HSSPIn_RXF:RFUS interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RFUS interrupt flag triggers the RX interrupt signal
[2]	RFOE	RX-FIFO Overrun Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RFOS interrupt flag is routed on the RX interrupt signal '0': The HSSPIn_RXF:RFOS interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RFOS interrupt flag triggers the RX interrupt signal
[1]	RFEE	RX-FIFO Empty Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RFES interrupt flag is routed on the RX interrupt signal. '0': The HSSPIn_RXF:RFES interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RFES interrupt flag triggers the RX interrupt signal
[0]	RFFE	RX-FIFO Full Interrupt Enable This bit decides whether or not the HSSPIn_RXF:RFFS interrupt flag is routed on the RX interrupt signal. '0': The HSSPIn_RXF:RFFS interrupt flag does not trigger the RX interrupt signal '1': The HSSPIn_RXF:RFFS interrupt flag triggers the RX interrupt signal

33.2.8 HSSPI RX Interrupt Clear Register (HSSPIn_RXC)

The HSSPI RX Interrupt Clear Register is used to clear the Interrupt Flags set in the HSSPIn_RXF register. By writing a '1' to a bit in this register, the software can clear the corresponding flag in the HSSPIn_RXF register.

HSSPI RX Interrupt Clear Register (HSSPIn_RXC)

Figure 33-9. HSSPI RX Interrupt Clear Register (HSSPIn_RXC)

HSSPIn_RXC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RSSRC	RFMTC	RFLETC	RFUC	RFOC	RFEC	RFFC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-9. HSSPI RX Interrupt Clear Register (HSSPIn_RXC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	RSSRC	Slave Select Released Interrupt Clear This bit is used to clear the HSSPIn_RXF:RSSRS interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RSSRS interrupt flag Read returns '0'.
[5]	RFMTC	RX-FIFO Fill Level More Than Threshold Interrupt Clear This bit is used to clear the HSSPIn_RXF:RFMTS interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RFMTS interrupt flag Read returns '0'.
[4]	RFLETC	RX-FIFO Fill Level Less Than or Equal to Threshold Interrupt Clear This bit is used to clear the HSSPIn_RXF:RFLETS interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RFLETS interrupt flag Read returns '0'.

Table 33-9. HSSPI RX Interrupt Clear Register (HSSPIn_RXC) bits

Bit Position	Bit Field Name	Bit Description
[3]	RFUC	RX-FIFO Underrun Interrupt Clear This bit is used to clear the HSSPIn_RXF:RFUS interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RFUS interrupt flag Read returns '0'.
[2]	RFOC	RX-FIFO Overrun Interrupt Clear This bit is used to clear the HSSPIn_RXF:RFOS interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RFOS interrupt flag Read returns '0'.
[1]	RFEC	RX-FIFO Empty Interrupt Clear This bit is used to clear the HSSPIn_RXF:RFES interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RFES interrupt flag Read returns '0'.
[0]	RFFC	RX-FIFO Full Interrupt Clear This bit is used to clear the HSSPIn_RXF:RFFS interrupt flag. '0': No effect '1': Clears the HSSPIn_RXF:RFFS interrupt flag Read returns '0'.

33.2.9 HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF)

The HSSPI Fault Interrupt Flag Register indicates the status of the fault interrupt flags. In the FCR4 Cluster Series, these interrupts are not propagated to the Interrupt Controller. All fault interrupt flags in this register are set as the result of a bus access. Whenever one of these flags is set, the corresponding bus access returns with a bus error response.

HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF)

Figure 33-10. HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF)

HSSPIn_FAULTF																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DRCBSFS	DWCBSFS	PVFS	WAFS	UMAFS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-10. HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4]	DRCBSFS	DMA Read Channel Block Size Fault This interrupt flag indicates that the block size fault has occured on a DMA read channel. The DMA read channel block size fault occurs if the HSSPI RX block counter is '0' and there is a valid read access to the RX FIFO (except from the DAP controller). This interrupt flag is non-maskable in the HSSPI module.
[3]	DWCBSFS	DMA Write Channel Block Size Fault This interrupt flag indicates that the block size fault has occured on a DMA write channel. The DMA write channel block size fault occurs if the HSSPI TX block counter is '0' and there is a valid write access to the TX FIFO. This interrupt flag is non-maskable in the HSSPI module.
[2]	PVFS	Protection Violation Fault This interrupt flag indicates that a Peripheral Protection Unit (PPU) protection violation fault has occurred. This interrupt flag is nonmaskable in the HSSPI module.

Table 33-10. HSSPI Fault Interrupt Flag Register (HSSPIn_FAULTF) bits

Bit Position	Bit Field Name	Bit Description
[1]	WAFS	Write Access Fault This interrupt flag indicates that a write access fault has occurred. This interrupt flag is non-maskable in the HSSPI module. This bit is set in command sequencer mode if HSSPIn_CSCFG:SRAM = '0' and an AHB master performs a write access to a memory location mapped onto the HSSPI memory area.
[0]	UMAFS	Unmapped Memory Access Fault This interrupt flag indicates that an unmapped memory access fault has occurred. This interrupt flag is non maskable in the HSSPI module. This bit is set by the HSSPI when any of the following conditions occur: a. In direct mode (i.e. HSSPIn_MCTRL:CSEN = '0'), an AHB access within the 256 MB address range starting from the HS_SPI base address is detected. b. In command sequencer mode (i.e. HSSPIn_MCTRL:CSEN = '1'), an AHB access to a memory device which is not enabled (in HSSPIn_CSCFG:SSEL0EN~SSEL3EN bits) is detected. c. In command sequencer mode (i.e. HSSPIn_MCTRL:CSEN = '1'), an AHB access to a memory location which is outside the memory range being mapped onto the four slave selects (configured through the HSSPIn_CSCFG:MSEL field) is detected. d. While the module is disabled (i.e. HSSPIn_MCTRL:MEN = '0'), an AHB access to a mapped memory is detected.

33.2.10 HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC)

The HSSPI Fault Interrupt Clear Register is used to clear the interrupt flags set in the HSSPIn_FAULTF register. By writing '1' to a bit in this register, the software can clear the corresponding flag in the HSSPIn_FAULTF register.

HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC)

Figure 33-11. HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC)

HSSPIn_FAULTC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DRCBSFC	DWCBSFC	PVFC	WAFC	UMAFC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-11. HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:5]	read0	-
[4]	DRCBSFC	DMA Read Channel Block Size Fault Interrupt Clear This bit is used to clear the HSSPIn_FAULTF:DRCBSFS interrupt flag. '0': No effect '1': Clears the HSSPIn_FAULTF:DRCBSFS interrupt flag Read returns '0'.
[3]	DWCBSFC	DMA Write Channel Block Size Fault Interrupt Clear This bit is used to clear the HSSPIn_FAULTF:DWCBSFS interrupt flag. '0': No effect '1': Clears the HSSPIn_FAULTF:DWCBSFS interrupt flag Read returns '0'.
[2]	PVFC	Protection Violation Fault Interrupt Clear This bit is used to clear the HSSPIn_FAULTF:PVFS interrupt flag. '0': No effect '1': Clears the HSSPIn_FAULTF:PVFS interrupt flag Read returns '0'.

Table 33-11. HSSPI Fault Interrupt Clear Register (HSSPIn_FAULTC) bits

Bit Position	Bit Field Name	Bit Description
[1]	WAFC	Write Access Fault Interrupt Clear This bit is used to clear the HSSPIn_FAULTF:WAFS interrupt flag. '0': No effect '1': Clears the HSSPIn_FAULTF:WAFS interrupt flag Read returns '0'.
[0]	UMAFC	Unmapped Memory Access Fault Interrupt Clear This bit is used to clear the HSSPIn_FAULTF:UMAFS interrupt flag. '0': No effect '1': Clears the HSSPIn_FAULTF:UMAFS interrupt flag Read returns '0'.

33.2.11 HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG)

The HSSPI Direct Mode Configuration Register configures the following operational parameters of the HSSPI:

- The master or slave mode of operation
- Software flow control or byte counter mode of slave select deassertion
- Enabling the iMSTART input signal triggers the initiation of a transfer

This register is used only when HSSPI module is in direct mode

HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG)

Figure 33-12. HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG)

HSSPIn_DMCFG							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	MSTARTEN	SSDC	MST
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	1

Table 33-12. HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG) bits

Bit Position	Bit Field Name	Bit Description
[7:3]	read0	-
[2]	MSTARTEN	iMSTART Enable This bit is used only when the HSSPI acts as a master (i.e. HSSPIn_DMCFG:MST = '1') in direct mode (i.e. HSSPIn_MCTRL:CSEN = '0'). '0': The HSSPI can initiate a transfer only when the software writes a '1' to the HSSPIn_DMSTART:START bit '1': The HSSPI can initiate a transfer either when the software writes a '1' to the HSSPIn_DMSTART:START bit or when a positive edge is detected on the iMSTART input signal of the HSSPI

Table 33-12. HSSPI Direct Mode Configuration Register (HSSPIn_DMCFG) bits

Bit Position	Bit Field Name	Bit Description
[1]	SSDC	Slave Select Deassertion Control This bit is used only when the HSSPI acts as a master (i.e. HSSPIn_DMCFG:MST = '1'), in which case it also decides how the slave select is de-asserted. '0': Software flow control. HSSPIn_DMSTOP:STOP is used to decide when to de-assert the slave select '1': Byte counter mode. HSSPIn_DMBCC:BCC is used to decide when to de-assert the slave select
[0]	MST	Master Mode '0': The HSSPI is in slave mode '1': The HSSPI is in master mode

33.2.12 HSSPI Direct Mode DMA Enable Register (HSSPIn_DMDMAEN)

The HSSPI direct mode DMA Enable Register can be used by the software, to enable/disable of the DMA service requests generated by the HSSPI. This register is used only when the HSSPI module is in direct mode.

HSSPI Direct Mode DMA Enable Register (HSSPIn_DMDMAEN)

Figure 33-13. HSSPI Direct Mode DMA Enable Register (HSSPIn_DMDMAEN

HSSPIn_DMDMAEN							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	TXDMAEN	RXDMAEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 33-13. HSSPI Direct Mode DMA Enable Register (HSSPIn_DMDMAEN) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	TXDMAEN	TX DMA Enable '0': TX DMA channel is disabled '1': TX DMA channel is enabled
[0]	RXDMAEN	RX DMA Enable '0': RX DMA channel is disabled '1': RX DMA channel is enabled

33.2.13 HSSPI Direct Mode Start Register (HSSPIn_DMSTART)

The HSSPI Direct Mode Start Register can be used by the software, to trigger the start of the serial transfer, if the HSSPI is working as a master. The HSSPIn_DMSTART:START bit can be set by the HSSPI while the HSSPIn_DMCFG:MSTARTEN bit is set and if a positive transition is detected on the iMSTART input signal. This register is used only when the HSSPI module is working as a master in direct mode.

HSSPI Direct Mode Start Register (HSSPIn_DMSTART)

Figure 33-14. HSSPI Direct Mode Start Register ((HSSPIn_DMSTART)

HSSPIn_DMSTART							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	START
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp1
0	0	0	0	0	0	0	0

Table 33-14. HSSPI Direct Mode Start Register ((HSSPIn_DMSTART) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	START	Start Transfer This field is used only when the HSSPI is configured as master (i.e. HSSPIn_DMCFG:MST = '1'). '0': No effect '1': Sets this bit The HSSPI resets this bit to '0' when it starts the serial transfer. Writing a '1' to this bit when the bit is already set to '1' has no effect on the current serial transfer. The HSSPI sets this bit to '1' if it is working as a master (i.e. HSSPIn_DMCFG:MST = '1') in direct mode (i.e. HSSPIn_MCTRL:CSEN = '0') and a positive edge is detected on iMSTART input signal (while HSSPIn_DMCFG:MSTARTEN is set).

33.2.14 HSSPI Direct Mode Stop Register (HSSPIn_DMSTOP)

The HSSPI Direct Mode Stop Register can be used by the software to stop the serial transfer if software flow control mode is selected (i.e. if HSSPIn_DMCFG:SSDC = '0'). This register is used only when the HSSPI module is working as a master in direct mode. Once the STOP bit is set, the software cannot clear this bit unless the HSSPI has stopped the current serial transfer (by de-asserting the slave select output).

HSSPI Direct Mode Stop Register (HSSPIn_DMSTOP)

Figure 33-15. HSSPI Direct Mode Stop Register (HSSPIn_DMSTOP)

HSSPIn_DMSTOP							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	STOP
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0

Table 33-15. HSSPI Direct Mode Stop Register (HSSPIn_DMSTOP) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	STOP	STOP bit This bit is used only when HS_SPI acts as a master (i.e. HSSPIn_DMCFG:MST = '1') and when HSSPIn_DMCFG:SSDC = '0'. This bit is used in software flow control mode for de-assertion of the slave select output. '0': The HSSPI does not de-assert the slave select output '1': De-assertion of the slave select output by the HSSPI depends on the mode used: In TX only mode - STOP is set and all content in HSSPIn_TXFIFO registers is transferred. In RX only mode - STOP is set only after the HSSPIn_RXFIFO registers and the HSSPIn_RXSHIFT register are full. In TX and RX mode - STOP is set and all content in HSSPIn_TXFIFO registers is transferred.

33.2.15 HSSPI Direct Mode Peripheral Select Register (HSSPIn_DMPSEL)

The HSSPI Direct Mode Peripheral Select Register can be used by the software to select one of the four peripheral slave select lines for initiating the serial transfer in master mode. This register is used only when the HSSPI module is in direct mode.

HSSPI Direct Mode Peripheral Slave Select Register (HSSPIn_DMPSEL)

Figure 33-16. HSSPI Direct Mode Peripheral Slave Select Register (HSSPIn_DMPSEL)

HSSPIn_DMPSEL							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	PSEL[1]	PSEL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 33-16. HSSPI Direct Mode Peripheral Slave Select Register (HSSPIn_DMPSEL) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1:0]	PSEL	Peripheral Select This bit is used only when the HSSPI acts as a master (i.e. HSSPIn_DMCFG:MST = '1'). The PSEL bits decide which of the 4 slave select output lines in SSEL3~0 is active for the current serial transfer. '00': Slave select 0 '01': Slave select 1 '10': Slave select 2 '11': Slave select 3

33.2.16 HSSPI Direct Mode Transfer Protocol Register (HSSPIn_DMTRP)

The HSSPI Direct Mode Transfer Protocol Register configures the transfer protocol of the serial transfer. This register is used only when the HSSPI module is in direct mode.

HSSPI Direct Mode Transfer Protocol Register (HSSPIn_DMTRP)

Figure 33-17. HSSPI Direct Mode Transfer Protocol Register (HSSPIn_DMTRP)

HSSPIn_DMTRP							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	TRP[3]	TRP[2]	TRP[1]	TRP[0]
Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 33-17. HSSPI Direct Mode Transfer Protocol Register (HSSPIn_DMTRP) bits

Bit Position	Bit Field Name	Bit Description
[7:4]	read0	-
[3:0]	TRP	Transfer Protocol Bits TRP[3:2] indicate duplex configuration - RX-only, TX-only or both: TX and RX. Bits TRP[1:0] indicate whether the protocol used is legacy, dual or quad. '0000': TX and RX in legacy mode '0100': RX only in legacy mode '0101': RX only in dual mode '0110': RX only in quad mode '1000': TX only in legacy mode '1001': TX only in dual mode '1010': TX only in quad mode All other combinations are reserved.

33.2.17 HSSPI Direct Mode Byte Count Control Register (HSSPIn_DMBCC)

The HSSPI Direct Mode Byte Count Control Register configures the number of bytes that should be transferred in the serial transfer if byte counter mode of flow control is selected (i.e. if HSSPIn_DMCFG:SSDC = '1'). This register is used only when the HSSPI module is configured to work as a master in direct mode.

HSSPI Direct Mode Byte Count Control Register (HSSPIn_DMBCC)

Figure 33-18. HSSPI Direct Mode Byte Count Control Register (HSSPIn_DMBCC)

HSSPIn_DMBCC															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BCC[15]	BCC[14]	BCC[13]	BCC[12]	BCC[11]	BCC[10]	BCC[9]	BCC[8]	BCC[7]	BCC[6]	BCC[5]	BCC[4]	BCC[3]	BCC[2]	BCC[1]	BCC[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-18. HSSPI Direct Mode Byte Count Control Register (HSSPIn_DMBCC) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BCC	Byte Count Control This field is used by the HSSPI only when it acts as a master (i.e. HSSPIn_DMCFG:MST = '1') in direct mode and when HSSPIn_DMCFG:SSDC = '1'. The BCC field must be programmed by the software with the number of bytes to be transmitted or received or both. The value in this field is loaded in a down counter at the start of a transfer and the counter is decremented when a byte is serially transferred. The HSSPI completes the transaction and de-asserts the slave select when this down counter reaches '0'. **Note:** The BCC value must be a multiple of the FIFO width (set in HSSPIn_FIFOCFG:FWIDTH) for all modes (Legacy, dual and quad protocols) which include RX

33.2.18 HSSPI Direct Mode Byte Count Status Register (HSSPIn_DMBCS)

The HSSPI Direct Mode Byte Count Status Register is a read-only register, which can be used by the software to know how many bytes are yet to be transferred in the current serial transfer. This register is valid only when the HSSPI module is configured to work as a master in direct mode and if byte counter mode of flow control is selected (i.e. if HSSPIn_DMCFG:SSDC = '1').

HSSPI Direct Mode Byte Count Status Register (HSSPIn_DMBCS)

Figure 33-19. HSSPI Direct Mode Byte Count Status Register (HSSPIn_DMBCS)

HSSPIn_DMBCS															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BCS[15]	BCS[14]	BCS[13]	BCS[12]	BCS[11]	BCS[10]	BCS[9]	BCS[8]	BCS[7]	BCS[6]	BCS[5]	BCS[4]	BCS[3]	BCS[2]	BCS[1]	BCS[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-19. HSSPI Direct Mode Byte Count Status Register (HSSPIn_DMBCS) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BCS	Byte Count Status This read-only field is valid only when the HSSPI acts as a master (i.e. HSSPIn_DMCFG:MST = '1') in direct mode and when HSSPIn_DMCFG:SSDC = '1'. The BCS field indicates the number of bytes in the current serial transfer that have not yet been serially transmitted, received or both.

33.2.19 HSSPI Direct Mode Status Register (HSSPIn_DMSTATUS)

The HSSPI Direct Mode Status Register contains the status bits, such as whether the TX/RX path is active/idle and the current fill-level of the TX/RX-FIFOs. This register is used only when the HSSPI module is configured in direct mode.

HSSPI Direct Mode Status Register (HSSPIn_DMSTATUS)

Figure 33-20. HSSPI Direct Mode Status Register (HSSPIn_DMSTATUS)

HSSPIn_DMSTATUS																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TXFLEVEL[4]	TXFLEVEL[3]	TXFLEVEL[2]	TXFLEVEL[1]	TXFLEVEL[0]	read0	read0	read0	RXFLEVEL[4]	RXFLEVEL[3]	RXFLEVEL[2]	RXFLEVEL[1]	RXFLEVEL[0]	read0	read0	read0	read0	read0	read0	TXACTIVE	RXACTIVE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-20. HSSPI Direct Mode Status Register (HSSPIn_DMSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:21]	read0	-
[20:16]	TXFLEVEL	Current Fill Level of TX-FIFO This field indicates the current fill level of the TX-FIFO.
[15:13]	read0	-
[12:8]	RXFLEVEL	Current Fill Level of RX-FIFO This field indicates the current fill level of the RX-FIFO.
[7:2]	read0	-
[1]	TXACTIVE	TX Active It indicates whether transmission is in progress. '0': Serial transmission is not active '1': Serial transmission is active
[0]	RXACTIVE	RX Active It indicates whether reception is in progress. '0': Serial reception is not active '1': Serial reception is active

33.2.20 HSSPI Transmit Bit Count Register (HSSPIn_TXBITCNT)

After a serial transfer halts or ends, the HSSPI Transmit Bit Count Register contains the number of pending transmission bits from the TX Shift Register. This register is used only when the HSSPI module is configured in direct mode.

HSSPI Transmit Bit Count Register (HSSPIn_TXBITCNT)

Figure 33-21. HSSPI Transmit Bit Count Register (HSSPIn_TXBITCNT)

HSSPIn_TXBITCNT							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	TXBITCNT[5]	TXBITCNT[4]	TXBITCNT[3]	TXBITCNT[2]	TXBITCNT[1]	TXBITCNT[0]
Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 33-21. HSSPI Transmit Bit Count Register (HSSPIn_TXBITCNT) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5:0]	TXBITCNT	TX Bit Count It indicates the number of bits pending transmission from the TX shift register. 0: No bits will be transmitted 1: One (1) bit will be transmitted ... 31: 31 bits will be transmitted 32: 32 bits will be transmitted The HSSPIn_TXBITCNT register is updated by the HSSPI when a transfer stops (while the HSSPI is in master mode) or when a transfer ends (i.e. slave select de-assertion interrupt flag is asserted).

33.2.21 HSSPI Receive Bit Count Register (HSSPIn_RXBITCNT)

After a serial transfer ends, the HSSPI Receive Bit Count Register contains the number of bits received in the RX Shift Register, which are not pushed by the HSSPI into the RX- FIFO. This register is used only when HSSPI module is configured in slave mode.

HSSPI Receive Bit Count Register (HSSPIn_RXBITCNT)

Figure 33-22. HSSPI Receive Bit Count Register (HSSPIn_RXBITCNT)

HSSPIn_RXBITCNT							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	RXBITCNT[5]	RXBITCNT[4]	RXBITCNT[3]	RXBITCNT[2]	RXBITCNT[1]	RXBITCNT[0]
Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 33-22. HSSPI Receive Bit Count Register (HSSPIn_RXBITCNT) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5:0]	RXBITCNT	RX Bit Count It indicates the number of valid bits in the RX shift register. 0: No bits are valid 1: One (1) bit is valid ... 31: 31 bits are valid 32: 32 bits are valid 'This register is used in slave mode only'. When a transfer ends in slave mode (i.e. slave select de-assertion interrupt flag is asserted), the HSSPIn_RXSHIFT register is updated with the assembled data and the HSSPIn_RXBITCNT register is updated with the number of valid bits in the HSSPIn_RXSHIFT register. The software can read the HSSPIn_RXSHIFT and HSSPIn_RXBITCNT registers to get the RX data which has not yet been pushed into the RX-FIFO.

33.2.22 HSSPI RX Shift Register (HSSPIn_RXSHIFT)

The HSSPI RX Shift Register is a read-only register. When a serial transfer in direct mode halts or ends, the data received by the HSSPI over the serial interface (which is not yet pushed into the RX-FIFO) is stored in this register. This register is used in slave mode only.

HSSPI RX Shift Register (HSSPIn_RXSHIFT)

Figure 33-23. HSSPI RX Shift (HSSPIn_RXSHIFT)

HSSPIn_RXSHIFT																															
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**
RXSHIFT[31]	RXSHIFT[30]	RXSHIFT[29]	RXSHIFT[28]	RXSHIFT[27]	RXSHIFT[26]	RXSHIFT[25]	RXSHIFT[24]	RXSHIFT[23]	RXSHIFT[22]	RXSHIFT[21]	RXSHIFT[20]	RXSHIFT[19]	RXSHIFT[18]	RXSHIFT[17]	RXSHIFT[16]	RXSHIFT[15]	RXSHIFT[14]	RXSHIFT[13]	RXSHIFT[12]	RXSHIFT[11]	RXSHIFT[10]	RXSHIFT[9]	RXSHIFT[8]	RXSHIFT[7]	RXSHIFT[6]	RXSHIFT[5]	RXSHIFT[4]	RXSHIFT[3]	RXSHIFT[2]	RXSHIFT[1]	RXSHIFT[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-23. HSSPI RX Shift Register (HSSPIn_RXSHIFT) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	RXSHIFT	RX Shift Register 'This register is used in slave mode only'. When a transfer ends in slave mode (i.e. slave select line is deasserted), the HSSPIn_RXSHIFT register is updated with the assembled data and the HSSPIn_RXBITCNT register is updated with the number of valid bits in the HSSPIn_RXSHIFT register. The software can read the HSSPIn_RXSHIFT and HSSPIn_RXBITCNT registers to get the RX data which has not yet been pushed into the RX-FIFO.

33.2.23 HSSPI TX-FIFO Registers (HSSPIn_TXFIFO0~15)

The HSSPI TX-FIFO Registers are used to push the data into the TX-FIFO. There are 16 such registers which are consecutively placed in the register map. Each of these 16 registers is identical in function. This is because AHB protocol does not support burst transfers to the same address. Only one register, HSSPIn_TXFIFO0 is explained here. Only 32-bit accesses are allowed to these registers. An 8-bit or a 16-bit access to these registers triggers a protection violation fault. A write access to these registers by the software pushes 33 bits into the TX-FIFO. The 33rd bit (i.e. the CTRL bit) is written with the value of the HSSPIn_FIFOCFG:TXCTRL bit.

HSSPI TX-FIFO Register 0 (HSSPIn_TXFIFO0)

Figure 33-24. HSSPI TX-FIFO Register 0 (HSSPIn_TXFIFO0)

HSSPIn_TXFIFO0																															
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**
TXDATA[31]	TXDATA[30]	TXDATA[29]	TXDATA[28]	TXDATA[27]	TXDATA[26]	TXDATA[25]	TXDATA[24]	TXDATA[23]	TXDATA[22]	TXDATA[21]	TXDATA[20]	TXDATA[19]	TXDATA[18]	TXDATA[17]	TXDATA[16]	TXDATA[15]	TXDATA[14]	TXDATA[13]	TXDATA[12]	TXDATA[11]	TXDATA[10]	TXDATA[9]	TXDATA[8]	TXDATA[7]	TXDATA[6]	TXDATA[5]	TXDATA[4]	TXDATA[3]	TXDATA[2]	TXDATA[1]	TXDATA[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-24. HSSPI TX-FIFO Register 0 (HSSPIn_TXFIFO0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TXDATA	TX-FIFO Register 0 Writing to this register puts the data into the next location of TX-FIFO and increments the TX-FIFO write pointer. Bit 33 of the TX-FIFO word is written with the value in the HSSPIn_FIFOCFG:TXCTRL field. Irrespective of the configured width of TX-FIFO, only 32-bit access to this register is allowed. When the configured TX-FIFO width is less than 32 bits, the unused most significant bits from the FIFO locations are not transmitted by the HSSPI, e.g. if configured FIFO width is 8 bits, then only the bits TXDATA[7:0] are transmitted by HS_SPI and the bits TXDATA[31:08] are unused. The software must not write any valid data to be transmitted in those unused most significant bits. A write access to this register while the TX-FIFO is full pushes the new data into the TX-FIFO and triggers a TX-FIFO overrun event. When a TX-FIFO overrun condition occurs, the integrity of the data transmitted over the serial lines is not guaranteed. Therefore, to avoid an overrun the software must ensure that TX-FIFO is not full before writing to it. **Note:** The relevant data is always right-aligned. This is also applicable for the last FIFO entry that is not an entire shift register width.

33.2.24 HSSPI RX-FIFO Registers (HSSPIn_RXFIFO0~15)

The HSSPI RX-FIFO Registers are used to pop the data out of the RX-FIFO. There are 16 such registers which are consecutively placed in the register map. Each of these 16 registers is identical in function. This is because the AHB protocol does not support burst transfers to the same address. Only one register (i.e. HSSPIn_RXFIFO0) is explained here. Only 32-bit read accesses are allowed to these registers. An 8-bit or a 16-bit access to these registers triggers a protection violation fault. By reading these register locations, the software can pop the data out of the RX-FIFO.

HSSPI RX-FIFO Register 0 (HSSPIn_RXFIFO0)

Figure 33-25. HSSPI RX-FIFO Register 0 (HSSPIn_RXFIFO0)

HSSPIn_RXFIFO0																															
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**
RXDATA[31]	RXDATA[30]	RXDATA[29]	RXDATA[28]	RXDATA[27]	RXDATA[26]	RXDATA[25]	RXDATA[24]	RXDATA[23]	RXDATA[22]	RXDATA[21]	RXDATA[20]	RXDATA[19]	RXDATA[18]	RXDATA[17]	RXDATA[16]	RXDATA[15]	RXDATA[14]	RXDATA[13]	RXDATA[12]	RXDATA[11]	RXDATA[10]	RXDATA[9]	RXDATA[8]	RXDATA[7]	RXDATA[6]	RXDATA[5]	RXDATA[4]	RXDATA[3]	RXDATA[2]	RXDATA[1]	RXDATA[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-25. HSSPI RX-FIFO Register 0 (HSSPIn_RXFIFO0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	RXDATA	RX-FIFO Register 0 Reading this register returns a word of data from the RX-FIFO location pointed to by the RX-FIFO read pointer. After a read access to this register, the RX-FIFO read pointer is incremented provided that the read cycle was initiated by the AHB master rather than the Debug Access Port (DAP) controller. If the DAP controller reads this register, the RX-FIFO read pointer is not incremented. A read access by a non-DAP AHB master to this register also updates the RX-FIFO read pointer. Therefore, non-DAP access in non-privileged mode can only be done with full access rights (i.e. When iPPU_ACCESS is '1'), else protection violation fault is trigerred. Irrespective of the configured width of RX-FIFO, only 32-bit read access to this register is allowed. When the configured FIFO width is less than 32 bits, the unused most significant bits from the FIFO locations contain invalid data, e.g. if the configured FIFO width is 8 bits, then only bits RXDATA[7:0] are valid and bits RXDATA[31:8] return logic '0'. The software must not use any data from the unused most significant bits. A read access to this register while the RX-FIFO is empty pops invalid data out of the RX-FIFO. A RX-FIFO underrun interrupt (HSSPIn_RXF:RFUS) event is triggered if the read cycle was initiated by the AHB master other than the DAP controller. If the DAP controller reads this register while the RX-FIFO is empty, the HSSPIn_RXF:RFUS flag is not set. **Note:** The relevant data is always right-aligned. However, this is not applicable in cases where the received data is less than the shift register width.

33.2.25 HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG)

The HSSPI FIFO Configuration Register configures the operation of the TX-FIFO and the RX-FIFO. The software can configure the FIFO threshold levels and width. The software can also flush the FIFOs using the TXFLSH and RXFLSH bits in this register.

HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG)

Figure 33-26. HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG)

HSSPIn_FIFOCFG																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TXFLSH	RXFLSH	TXCTRL	FWIDTH[1]	FWIDTH[0]	TXFTH[3]	TXFTH[2]	TXFTH[1]	TXFTH[0]	RXFTH[3]	RXFTH[2]	RXFTH[1]	RXFTH[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1

Table 33-26. HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:13]	read0	-
[12]	TXFLSH	TX-FIFO Flush This register can be used by the software to flush the TX-FIFO. '0': No effect '1': Flushes the TX-FIFO Read returns a '0'.
[11]	RXFLSH	RX-FIFO Flush This register can be used by the software to flush the RX-FIFO. '0': No effect '1': Flushes the RX-FIFO Read returns a '0'.

Table 33-26. HSSPI FIFO Configuration Register (HSSPIn_FIFOCFG) bits

Bit Position	Bit Field Name	Bit Description
[10]	TXCTRL	TXCTRL bit to be Written to TX-FIFO When a write to the HSSPIn_TXFIFO0~15 register occurs, bit 33 in the TX-FIFO word is written with the value in the HSSPIn_FIFOCFG:TXCTRL field. The HSSPIn_FIFOCFG:TXCTRL should be set by the software only when the HSSPIn_DMTRP:TRP field is programmed in any one of the following modes: a. TX only in dual mode b. TX only in quad mode If the HSSPIn_DMTRP:TRP field is programmed in any mode other than the two modes mentioned above, then the TXCTRL bit shall be programmed to '0' by the software. Before writing to the HSSPIn_TXFIFO0~15 register, the software must update this bit, depending on whether it wants the TXCTRL bit in the next location in TX-FIFO to be set or reset.
[9:8]	FWIDTH	FIFO Width This register field indicates the FIFO width. Depending on the configured width of the FIFO, the usable size of the shift register in the SPI core also changes. '00': TX-FIFO, RX-FIFO and shift register are 8-bit wide '01': TX-FIFO, RX-FIFO and shift register are 16-bit wide '10': TX-FIFO, RX-FIFO and shift register are 24-bit wide '11': TX-FIFO, RX-FIFO and shift register are 32-bit wide
[7:4]	TXFTH	TX-FIFO Threshold Level Software must program this field with the threshold level of the TX-FIFO.
[3:0]	RXFTH	RX-FIFO Threshold Level Software must program this field with the threshold level of the RX-FIFO.

33.2.26 HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG)

The HSSPI Command Sequencer Configuration Register configures the command sequencer in the HSSPI module. This register must be programmed by the software before enabling the command sequencer mode. Attributes such as transfer protocol, memory write enable/disable and size of the memory device interfaced with HSSPI can be configured in this register.

HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG)

Figure 33-27. HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG)

HSSPIn_CSCFG																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	MSEL[3]	MSEL[2]	MSEL[1]	MSEL[0]	read0	read0	read0	read0	SSEL3EN	SSEL2EN	SSEL1EN	SSEL0EN	read0	read0	read0	read0	read0	MBM[1]	MBM[0]	SRAM
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-27. HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:20]	read0	-
[19:16]	MSEL	Memory Device Selection bits This field indicates the range of the AHB address space associated with each slave select line. It also indicates the size of each memory banks in the selected device. This field is used by the command sequencer for two things: a. To select which of the 4 slave select output lines should be asserted for the memory mapped serial transfer b. To select the size of each memory bank in the selected memory device. For more details refer to 33.5 Command sequencer mode.
[15:12]	read0	-
[11]	SSEL3EN	Slave Select 3 Enable '0': Any access which falls into the memory range mapped onto slave select 3 causes an unmapped memory access fault '1': Access to the serial memory device mapped onto slave select 3 is enabled

Table 33-27. HSSPI Command Sequencer Configuration Register (HSSPIn_CSCFG) bits

Bit Position	Bit Field Name	Bit Description
[10]	SSEL2EN	Slave Select 2 Enable '0': Any access which falls into the memory range mapped onto slave select 2 causes an unmapped memory access fault '1': Access to the serial memory device mapped onto slave select 2 is enabled
[9]	SSEL1EN	Slave Select 1 Enable '0': Any access which falls into the memory range mapped onto slave select 1 causes an unmapped memory access fault '1': Access to the serial memory device mapped onto slave select 1 is enabled
[8]	SSEL0EN	Slave Select 0 Enable '0': Any access which falls into the memory range mapped onto slave select 0 causes an unmapped memory access fault '1': Access to the serial memory device mapped onto slave select 0 are enabled
[7:3]	read0	-
[2:1]	MBM	Multi Bit Mode '00': Memory device access through the command sequencer uses the legacy SPI protocol. Read data is sampled on SDATA[0]. Memory instruction, address and other control information are transmitted on SDATA[1]. The output enables of the other serial data lines are de-asserted '01': Memory device access through the command sequencer uses the half-duplex dual-bit SPI protocol. Read data is sampled on SDATA[1:0]. Memory instruction, address and other control information are transmitted on SDATA[1:0] '10': Memory device access through the command sequencer uses the quad-bit SPI protocol. Read data is sampled on SDATA[3:0]. Memory instruction, address and other control information are transmitted on SDATA[3:0] '11': Reserved
[0]	SRAM	Serial SRAM or Serial Flash Memory Type Select This bit should be set only if serial SRAM devices are memory mapped through the HSSPI. '0': Serial Flash memory devices are connected. Writes are disabled '1': Serial SRAM memory devices are connected. Writes are enabled

33.2.27 HSSPI Command Sequencer Idle Time Register (HSSPIn_CSITIME)

The HSSPI Command Sequencer Idle Time Register configures the idle timeout period of the command sequencer in the HSSPI module. The software must program this timeout value before enabling the command sequencer mode.

HSSPI Command Sequencer Idle Time Register (HSSPIn_CSITIME)

Figure 33-28. HSSPI Command Sequencer Idle Time Register (HSSPIn_CSITIME)

HSSPIn_CSITIME																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ITIME[15]	ITIME[14]	ITIME[13]	ITIME[12]	ITIME[11]	ITIME[10]	ITIME[9]	ITIME[8]	ITIME[7]	ITIME[6]	ITIME[5]	ITIME[4]	ITIME[3]	ITIME[2]	ITIME[1]	ITIME[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 33-28. HSSPI Command Sequencer Idle Time Register (HSSPIn_CSITIME) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	ITIME	Idle Time This register is used by the HSSPI in command sequencer mode (i.e. HSSPIn_MCTRL:CSEN = '1') only. Once the HSSPI completes the required number of memory read or write accesses on the serial interface, it keeps the slave select line asserted. If no further access to the mapped serial memory device is detected within the idle timeout period, then the HSSPI deasserts the slave select line only after a further six (6) SCLK cycles. This gives better performance when the serial memory access is of the same type (i.e. if all are read or write accesses), the accessed locations are continuous and the access occurs within the predefined idle timeout interval. The idle timeout interval is in terms of the AHB clock period.

33.2.28 HSSPI Command Sequencer Address Extension Register (HSSPIn_CSAEXT)

The HSSPI Command Sequencer Address Extension Register is used to extend the usable size of memory mapped to the command sequencer. The software must program this register if the address extension feature to virtually access a serial memory of up to 16 GB is required. If address extension is not to be used, the software must reset all bits in this register to '0'.

HSSPI Command Sequencer Address Extension Register (HSSPIn_CSAEXT)

Figure 33-29. HSSPI Command Sequencer Address Extension Register (HSSPIn_CSAEXT)

HSSPIn_CSAEXT																														
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**
AEXT[18]	AEXT[17]	AEXT[16]	AEXT[15]	AEXT[14]	AEXT[13]	AEXT[12]	AEXT[11]	AEXT[10]	AEXT[9]	AEXT[8]	AEXT[7]	AEXT[6]	AEXT[5]	AEXT[4]	AEXT[3]	AEXT[2]	AEXT[1]	AEXT[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-29. HSSPI Command Sequencer Address Extension Register (HSSPIn_CSAEXT) bits

Bit Position	Bit Field Name	Bit Description
[31:13]	AEXT	Address Extension bits This register is used by the HSSPI in command sequencer mode (i.e. HSSPIn_MCTRL:CSEN = '1') only. The HSSPIn_CSAEXT register contains the 19 most significant bits [31:13] of the memory address which are generated by the command sequencer. The memory address generated by the HSSPI on each slave select (while in command sequencer mode) is a concatenation of the appropriate number of bits from the HSSPIn_CSAEXT register and from the AHB address bus. For more details refer to 33.5 Command sequencer mode. If address extension is not to be used, the software should reset this field to 0x0000000.
[12:8]	read0	-
[7:0]	read0	-

33.2.29 HSSPI Read Command Sequence Data/Control Register 0~7 (HSSPIn_RDCSDC0~7)

The HSSPI Read Command Sequence Data/Control Register 0~7 are a part of the list of eight Data/Control registers which configure the phases of the serial transaction generated by the command sequencer for memory read operations. These registers are used only in command sequencer mode. Only one of these registers (HSSPIn_RDCSDC0) is explained here. Other registers have the same fields.

HSSPI Read Command Sequence Data/Control Register 0 (HSSPIn_RDCSDC0)

Figure 33-30. HSSPI Read Command Sequence Data/Control Register 0 (HSSPIn_RDCSDC0)

HSSPIn_RDCSDC0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RDCSDATA[7]	RDCSDATA[6]	RDCSDATA[5]	RDCSDATA[4]	RDCSDATA[3]	RDCSDATA[2]	RDCSDATA[1]	RDCSDATA[0]	read0	read0	read0	read0	read0	read0	read0	DEC
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-30. HSSPI Read Command Sequence Data/Control Register 0 (HSSPIn_RDCSDC0) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	RDCSDATA	Command Sequencer Data or Control byte for Memory-Read Transactions '0': When the HSSPIn_RDCSDC0:DEC bit is '0', the RDCSDATA field contains the 8-bit data to be transmitted on the serial interface '1': When the HSSPIn_RDCSDC0:DEC bit is '1', the RDCS-DATA[2:0] field is decoded as follows '000': RDCSDATA[2:0] = '000': Transmit address bits [7:0] of the serial memory address '001': RDCSDATA[2:0] = '001': Transmit address bits [15:8] of the serial memory address '010': RDCSDATA[2:0] = '010': Transmit address bits [23:16] of the serial memory address '011': RDCSDATA[2:0] = '011': Transmit address bits [31:24] of the serial memory address '100': RDCSDATA[2:0] = '100': High-Z byte (i.e. SDATA[3:0] signals are tri-stated for 1 byte time '101': RDCSDATA[2:0] = '101': High-Z nibble (i.e. transmission of RDCSDATA[7:4] is followed by tri-stating of SDATA output for 1 nibble time '111': RDCSDATA[2:0] = '111': End of list All other values of RDCSDATA[2:0] are reserved and must not be used. **Limitations**: ■ The last command in a command sequence (i.e. the entry immediately before the "End of list" or the entry in the last register if the command sequence has a length of 8) must be either "Hi-Z byte" or "Hi-Z nibble". ■ "End of list", "Hi-Z byte" and "Hi-Z nibble" must not be the first entry on the list ■ The only commands that can follow "Hi-Z byte" to the "End of list" are "Hi-Z byte" and "End of list" ■ The only commands that can follow "Hi-Z nibble" to the "End of list" are "Hi-Z byte" and "End of list"
[7:1]	read0	-
[0]	DEC	Decode '0': Transmit HSSPIn_RDCSDC0:RDCSDATA as it is '1': Decode HSSPIn_RDCSDC0:RDCSDATA[2:0] to decide what further action is required

33.2.30 HSSPI Write Command Sequence Data/Control Register 0~7 (HSSPIn_WRCSDC0~7)

The HSSPI Write Command Sequence Data/Control Register 0~7 are part of the list of eight Data/Control registers which configure the phases of the serial transaction generated by the command sequencer for memory write operations. These registers are used only in command sequencer mode. Only one of these registers (HSSPIn_WRCSDC0) is explained here. Other registers have same fields.

HSSPI Write Command Sequence Data/Control Register 0 (HSSPIn_WRCSDC0)

Figure 33-31. HSSPI Write Command Sequence Data/Control Register 0 (HSSPIn_WRCSDC0)

HSSPIn_WRCSDC0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
WRCSDATA[7]	WRCSDATA[6]	WRCSDATA[5]	WRCSDATA[4]	WRCSDATA[3]	WRCSDATA[2]	WRCSDATA[1]	WRCSDATA[0]	read0	read0	read0	read0	read0	read0	read0	DEC
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33-31. HSSPI Write Command Sequence Data/Control Register 0 (HSSPIn_WRCSDC0) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	WRCSDATA	Command Sequencer Data or Control byte for Memory-Write Transactions '0': When the HSSPIn_WRCSDC0:DEC bit is '0', the WRCSDATA field contains the 8-bit data to be transmitted on the serial interface '1': When the HSSPIn_WRCSDC0:DEC bit is '1', the WRCS-DATA[2:0] field is decoded as follows '000': WRCSDATA[2:0] = '000': Transmit address bits [7:0] of the serial memory address '001': WRCSDATA[2:0] = '001': Transmit address bits [15:08] of the serial memory address '010': WRCSDATA[2:0] = '010': Transmit address bits [23:16] of the serial memory address '011': WRCSDATA[2:0] = '011': Transmit address bits [31:24] of the serial memory address '100': WRCSDATA[2:0] = '100': High-Z byte (i.e. SDATA[3:0] signals are tri-stated for 1 byte time '101': WRCSDATA[2:0] = '101': High-Z nibble (i.e. transmission of WRCSDATA[7:4] is followed by tri-stating of SDATA output for 1 nibble time '111': WRCSDATA[2:0] = '111': End of list All other values of WRCSDATA[2:0] are reserved and must not be used. **Limitations**: ■ "End of list", "Hi-Z byte" and "Hi-Z nibble" must not be the first entry on the list ■ The only commands that can follow "Hi-Z byte" to the "End of list" are "Hi-Z byte" and "End of list" ■ The only commands that can follow "Hi-Z nibble" to the "End of list" are "Hi-Z byte" and "End of list"
[7:1]	read0	-
[0]	DEC	Decode '0': Transmit HSSPIn_WRCSDC0:WRCSDATA as it is '1': Decode the HSSPIn_WRCSDC0:WRCSDATA[2:0] to decide what further action ir required

33.2.31 HSSPI Module ID Register (HSSPIn_MID)

This is a read-only register with a unique module identification number which identifies the version of the HSSPI module used in the MCU. Refer to the device datasheet for the module identification number of the HSSPI module in your device.

HSSPI Module ID Register (HSSPIn_MID)

Figure 33-32. HSSPI Module ID Register (HSSPIn_MID)

HSSPIn_MID																															
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**
MID[31]	MID[30]	MID[29]	MID[28]	MID[27]	MID[26]	MID[25]	MID[24]	MID[23]	MID[22]	MID[21]	MID[20]	MID[19]	MID[18]	MID[17]	MID[16]	MID[15]	MID[14]	MID[13]	MID[12]	MID[11]	MID[10]	MID[9]	MID[8]	MID[7]	MID[6]	MID[5]	MID[4]	MID[3]	MID[2]	MID[1]	MID[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 33-32. HSSPI Module ID Register (HSSPIn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID This read-only register gives the unique module identification (ID) number of the HSSPI module, which in turn identifies the version of the the HSSPI module used in the MCU. The module ID number of an HSSPI is found in the device specific datasheet.

33.3 Operation of the HSSPI

This chapter describes the operation of HSSPI.

33.3.1 Modes of operation

HSSPI can be configured in one of the two operating modes: direct mode and command sequencer mode.

Direct mode

In direct mode of operation the MCU can directly write the data to be transmitted into the TX-FIFO. Similarly the MCU can directly read the data received over the serial interface from the RX-FIFO and from the shift register. The SPI core transfers the data to/from the FIFOs over the serial interface. Based on the configuration in CSR in direct mode, the HSSPI can work either as an SPI master or as an SPI slave. The direct mode is described in 33.4 Direct mode.

Command sequencer mode

In command sequencer mode the HSSPI can act as an SPI master only. In this mode, the HSSPI maps the external serial Flash or serial SRAM devices to the address space of the MCU. Up to four serial memory devices can be mapped in this way, one on each of the four slave select outputs. If the MCU (or any other AHB master) initiates an AHB transfer to access any of the mapped serial memory device, the HSSPI initiates serial transfer for the corresponding memory read or write operation. Till such time as the HSSPI accesses the external device, the AHB transfer is stalled. The command sequencer mode is described in 33.5 Command sequencer mode.

Clocking modes

Based on the programmed values of the HSSPIn_PCC0~3:CPOL, HSSPIn_PCC0~3:CPHA and HSSPIn_PCC0~3:ACES bits, each peripheral can have up to eight clocking modes. These bits, along with the HSSPIn_PCC0~3:RTM bits, decide the serial data input and output timings of the HSSPI with respect to the serial SPI clock. This is explained in Table 33-33..

Table 33-33. Clocking modes

Mode	ACES (active clock edges are same)	CPOL (clock polarity)	CPHA (clock phase)	Description
Mode 0	'0'	'0'	'0'	Output data is driven one half-cycle before the first positive edge of the serial clock and on subsequent **negative** edges.
				Input data is sampled on **positive** edges of the serial clock.
Mode 1		'0'	'1'	Output data is driven on **positive** edges of the serial clock.
				Input data is sampled on the **negative** edges of the serial clock.
Mode 2		'1'	'0'	Output data is driven one half-cycle before the first negative edge of the serial clock and on subsequent **positive** edges.
				Input data is sampled on the **negative** edges of the serial clock.
Mode 3		'1'	'1'	Output data is driven on the **negative** edge of the serial clock.
				Input data is sampled on the **positive** edges.

Table 33-33. Clocking modes

Mode	ACES (active clock edges are same)	CPOL (clock polarity)	CPHA (clock phase)	Description
Mode 4	'1'	'0'	'0'	Output data is driven one half-cycle before the first positive edge of the serial clock and on subsequent **negative** edges.
				Input data is sampled on **negative** edges of the serial clock.
Mode 5		'0'	'1'	Output data is driven on **positive** edge of the serial clock.
				Input data is sampled one half-cycle after the first negative edge of the serial clock and on the subsequent **positive** edges of the serial clock.
Mode 6		'1'	'0'	Output data is driven one half-cycle before the first negative edge of the serial clock and on subsequent **positive** edges.
				Input data is sampled on the **positive** edges of the serial clock.
Mode 7		'1'	'1'	Output data is driven on the **negative** edge of the serial clock.
				Input data is sampled one half-cycle after the first positive edge of the serial clock and on the subsequent **negative** edges of the serial clock.

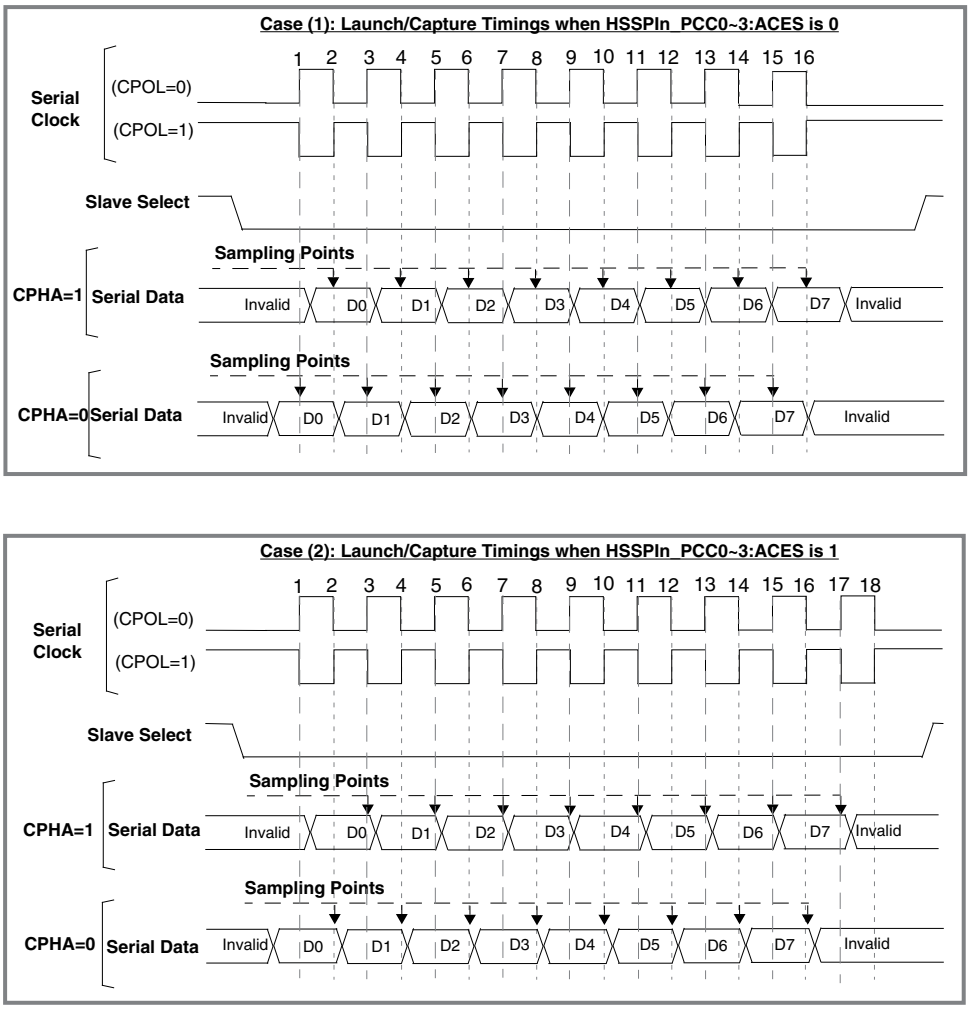
Timing waveforms, indicating the serial data and the serial clock, along with the different combinations of HSSPIn_PCC0~3:ACES, HSSPIn_PCC0~3:CPOL, and HSSPIn_PCC0~3:CPHA bits are depicted in Figure 33-33.

As indicated in Figure 33-33, when the HSSPIn_PCC0~3:ACES bit is set, the data driving and sampling points are separated by one complete clock period (as opposed to the case in the traditional HSSPIn_PCC0~3:ACES = '0' configuration, where the data driving and sampling points are separated only with a half-clock period). Thus, when HSSPIn_PCC0~3:ACES is set, the transfer runs for one extra clock cycle. Upon the of a transfer in receive mode, the HSSPI skips the sampling of data on the first sampling point and starts sampling the data from the next sampling point. This skipping of the data on the first sampling point is done in order to capture the correct serial data in retimed mode.

Note:

When HSSPIn_PCC0~3:ACES is set, if transmission and reception are both enabled simultaneously through the CSRs, the extra clock cycle inserted at the start of the reception has the side-effect of also transmitting some data for an extra clock cycle to the SPI slave which is interfaced with the HSSPI master. Therefore, to avoid this extra data transmission when HSSPIn_PCC0~3:ACES is configured, disable the transmission while reception is enabled.

Figure 33-33. Clocking modes of the serial interface clock



As shown in the figure, when HSSPIn_PCC0~3:ACES is set to '1', one extra clock cycle is required for the serial data to be correctly captured by the remote device.

33.3.2 Retimed clock

Some of the serial flash devices use SCLK frequencies of 80 MHz (and above), and they leave very tight setup margins for the serial data to be captured by the HSSPI. When the HSSPI is interfaced with such memory devices, data setup violations might occur in the registers, that are used to capture the serial data input. To capture the valid data read from such serial memories, the retimed clock mode should be used. Retimed clock mode can be set using the HSSPIn_PCC0~3:RTM bit.

Figure 33-34. Retimed clock in HSSPI

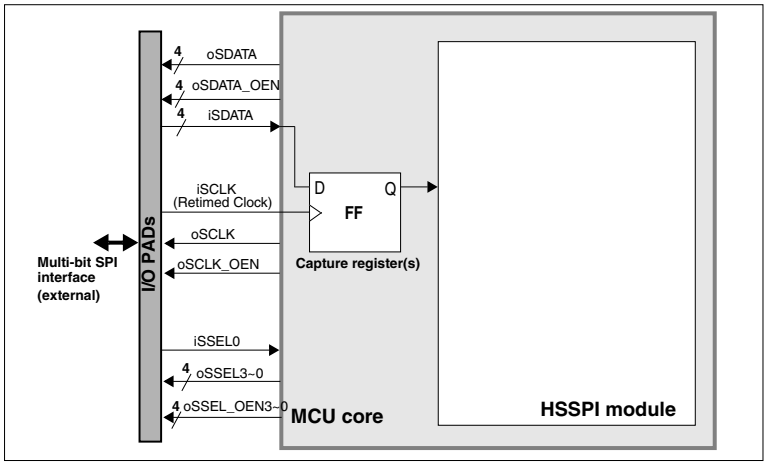


Figure 33-34 shows how the retimed clock is generated in this module. The registers used to capture the serial data input are placed physically close to the I/O pads in the MCU boundary. In retimed clock mode (i.e. when HSSPIn_PCC0~3:RTM = '1'), these registers are sourced from the clock input which is looped back to the HSSPI (from oSCLK to the I/O pad at the periphery of the MCU, and back to the HSSPI) i.e. iSCLK input is used for capturing the input data. This is called retimed clock. This retiming technique is a cycle stealing technique, which allows late arriving serial data signals (i.e. iSDATA[3:0]) to be sampled at a later point in time by intentionally introducing a skew on the clock.

33.3.3 Serial clock frequency

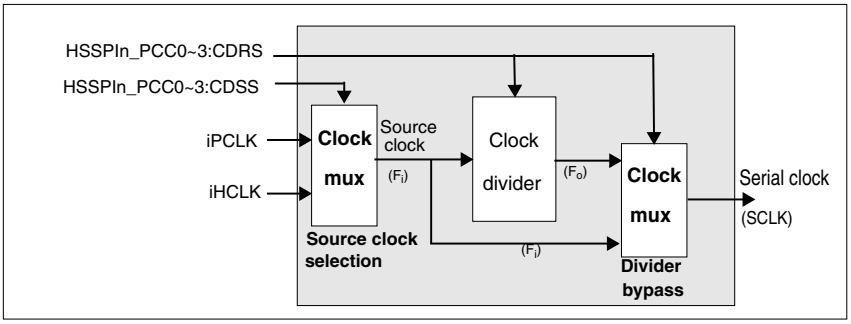
When the HSSPI is programmed in master mode, the SCLK clock is internally generated by dividing either the AHB clock (iHCLK) or the peripheral clock (iPCLK). The HSSPIn_PCC0~3 registers control the selection of the source clock for each of the four slave selects. The clock division ratio of the resulting internal clock divider can also be programmed in these registers. Figure 33-35 shows how the serial clock is generated. The HSSPIn_PCC0~3:CDRS field decides the clock division ratio. The frequency 'F_o' of the clock generated by the clock divider is given by the equation:

$F_o = F_i / (2 \times$ HSSPIn_PCC0~3:CDRS$)$

Where F_i is the frequency of the source clock selected by the HSSPIn_PCC0~3:CDSS field.

If the HSSPIn_PCC0~3:CDRS field is '0', the clock divider is bypassed, so that the frequency of SCLK output is same as that of source clock. For a non-zero value of HSSPIn_PCC0~3:CDRS field, the output of clock divider (i.e. clock with frequency F_o) is the serial clock (i.e. SCLK).

Figure 33-35. Serial clock generation in master mode of operation



33.3.4 SPI protocol

The HSSPI supports both the legacy SPI as well as the new dual-bit or quad-bit SPI protocol. While in direct mode of operation the HSSPIn_DMTRP:TRP[1:0] bits decide whether HSSPI uses the legacy, dual-bit or quad-bit protocol. While in command sequencer mode, the HSSPIn_CSCFG:MBM bits decide whether the HSSPI uses the legacy, dual-bit or quad-bit protocol. The dual-bit and quad-bit SPI protocol is used for interfacing with the newer generation serial flash memory devices.

Legacy SPI protocol

The legacy SPI protocol is a full-duplex protocol. When the HSSPI is configured in master mode with the legacy SPI protocol, the data can be received on a single wire (i.e. SDATA[1]) and simultaneously transmitted on a single wire (i.e. SDATA[0]). When HSSPI is configured in slave mode with legacy SPI protocol, the data can be received on a single wire (i.e. SDATA[0]) and simultaneously, the data can also be transmitted on a single wire (i.e. SDATA[1]). While legacy SPI protocol is being used, the unused data lines (i.e. SDATA[2] and SDATA[3]) are tri-stated by the HSSPI. In direct mode, when HSSPIn_DMTRP:TRP is configured for 'TX and RX in legacy mode', 'TX only in legacy mode' or 'RX only in legacy mode', the full-duplex legacy SPI protocol is used by HSSPI.

Dual bit protocol

In a dual-bit SPI protocol, two serial data lines (i.e. SDATA[1:0]) are used in a half-duplex manner. Data transmission and reception cannot happen simultaneously. While dual-bit SPI protocol is being used, the unused data lines (i.e. SDATA[2] and SDATA[3]) are tri-stated by the HSSPI. In direct mode, when HSSPIn_DMTRP:TRP is configured for 'TX only in dual mode', or 'RX only in dual mode', the dual-bit SPI protocol is used.

Quad bit protocol

In quad-bit SPI protocol all four serial data lines (i.e. SDATA[3:0]) are used in a half-duplex manner. Data transmission and reception cannot happen simultaneously. In direct mode, when HSSPIn_DMTRP:TRP is configured for 'TX only in quad mode' or 'RX only in quad mode', the quad-bit SPI protocol is used.

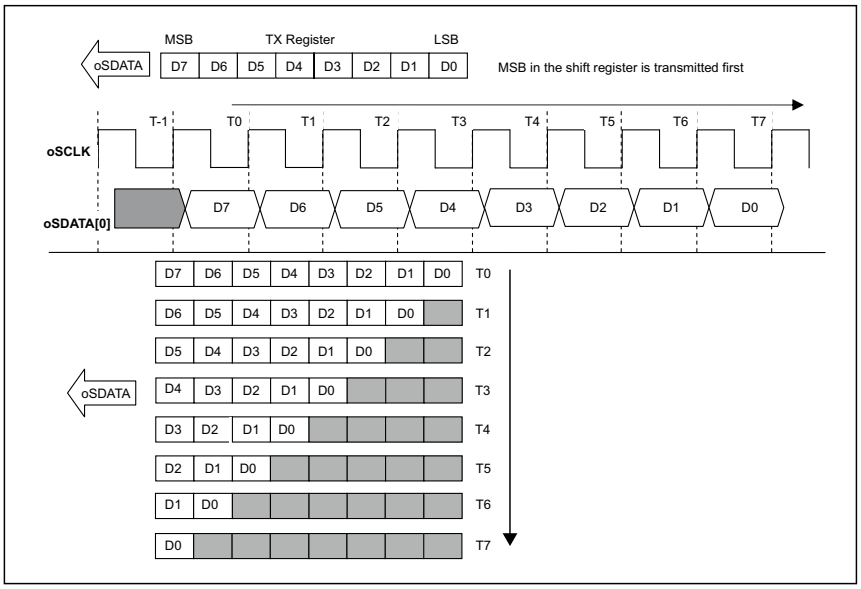
33.3.5 Shift direction

The HSSPI Peripheral Communication Configuration (HSSPIn_PCC0~3) registers have a bit (i.e. SDIR), which decides the direction in which the shift register is shifted.

When HSSPIn_PCC0~3:SDIR is '0', the most significant bit in the shift register is transmitted first and the data received first is shifted into the least significant bit in the shift register, i.e. the shift register is shifted left, as shown in Figure 33-36, Figure 33-37 and Figure 33-38 for legacy, dual and quad modes respectively for 8-bit shift registers. An example of a 32-bit register in quad mode is shown in Figure 33-39 (transmit) and Figure 33-40 (receive). All waveforms assume HSSPIn_PCC0~3:CPOL = '0', HSSPIn_PCC0~3:CPHA = '0' and HSSPIn_FIFOCFG:FWIDTH = '0'. When HSSPIn_PCC0~3:SDIR is '1', the least significant bit in the shift register is transmitted first and the data received first is shifted into the most significant bit in the shift register, i.e. the shift register is shifted right. Irrespective of the value of the HSSPIn_PCC0~3:SDIR bit, the read/write accesses to the data registers always have the least significant bit of the data in bit '0'.

Note: The source of transmitted data could also be the other data registers, like the HSSPIn_RDCSDC0~7:RDCSDATA or the HSSPIn_WRCSDC0~7:WRCSDATA.

Figure 33-36. Shift direction 8-bit register - Legacy mode (assumptions: HSSPIn_PCC0~3:CPOL = ’0’, HSSPIn_PCC0~3:SDIR = ’0’, HSSPIn_FIFOCFG:FWIDTH = ’0’)



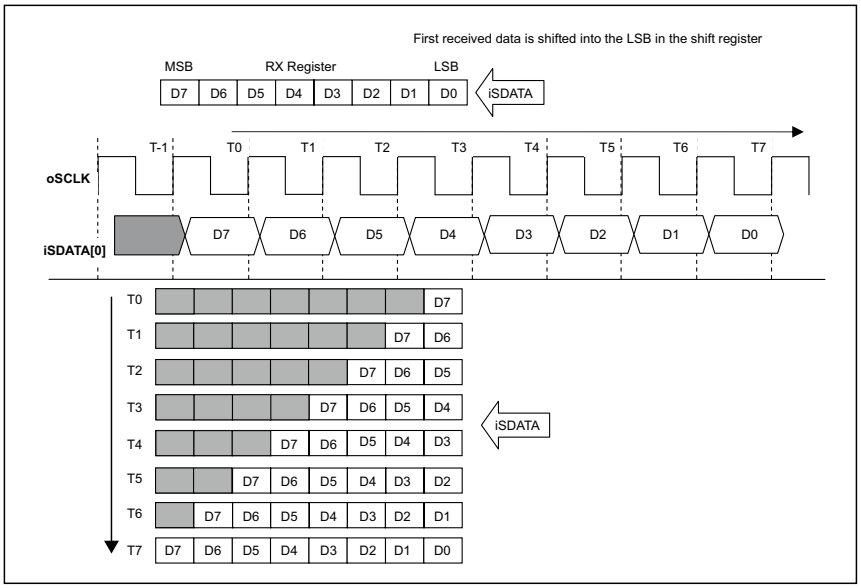
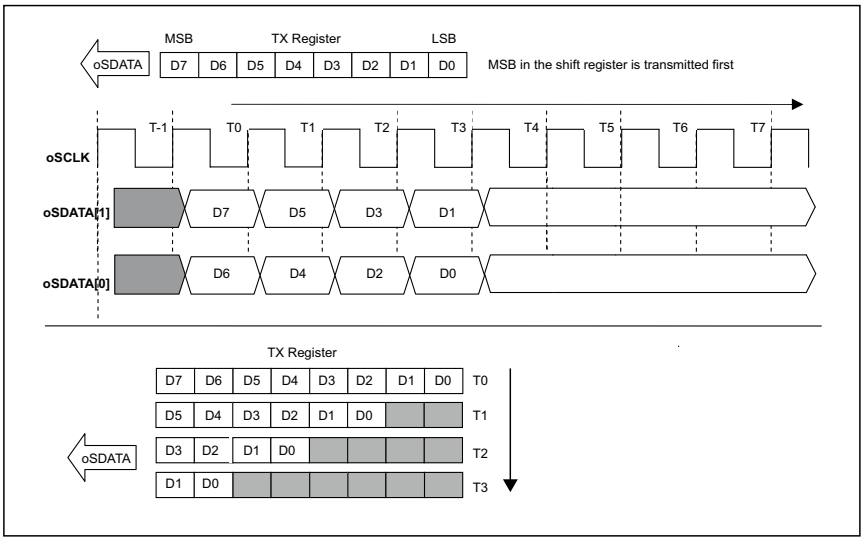


Figure 33-37. Shift direction for 8-bit registers - Dual mode (assumptions: HSSPIn_PCC0~3:CPOL = '0', HSSPIn_PCC0~3:CPHA = '0', HSSPIn_PCC0~3:SDIR = '0', HSSPIn_FIFOCFG:FWIDTH = '0')



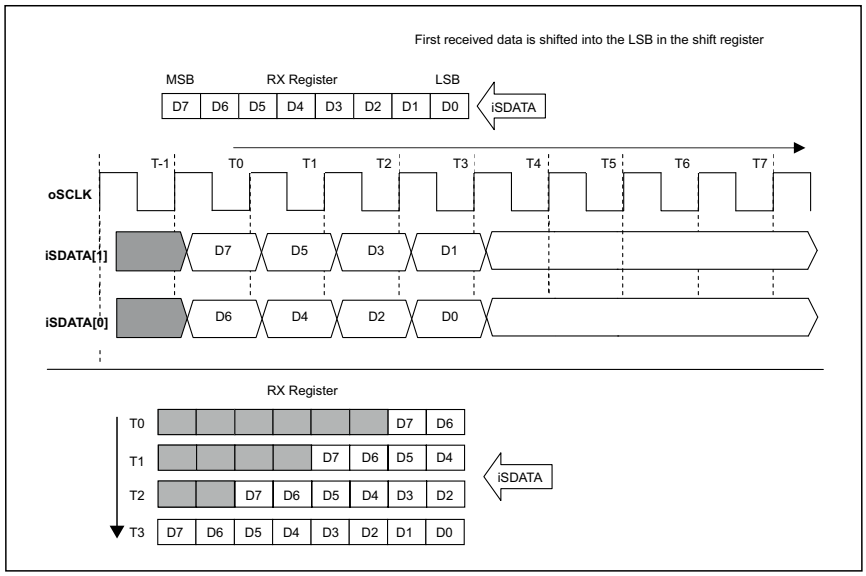
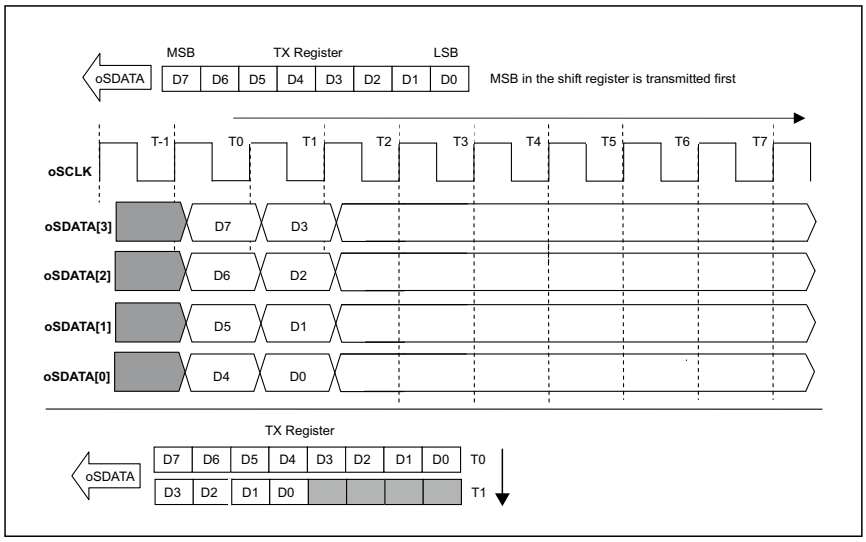


Figure 33-38. Shift direction for 8-bit registers - Quad mode (assumptions: HSSPIn_PCC0~3:CPOL = '0', HSSPIn_PCC0~3:CPHA = '0', HSSPIn_PCC0~3:SDIR = '0', HSSPIn_FIFOCFG:FWIDTH = '0')



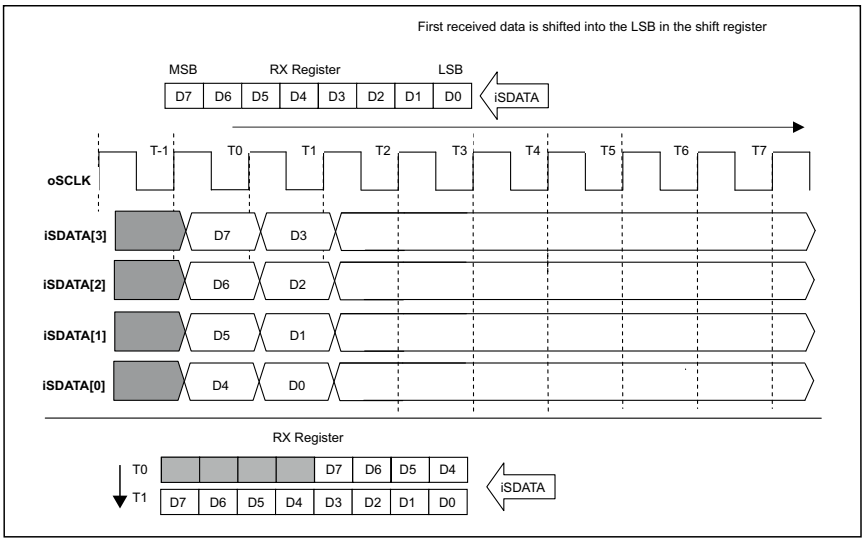


Figure 33-39. Shift direction for 32-bit registers - Quad mode TX (assumptions: HSSPIn_PCC0~3:CPOL = '0', HSSPIn_PCC0~3:CPHA = '0', HSSPIn_PCC0~3:SDIR = '0', HSSPIn_FIFOCFG:FWIDTH = '11')

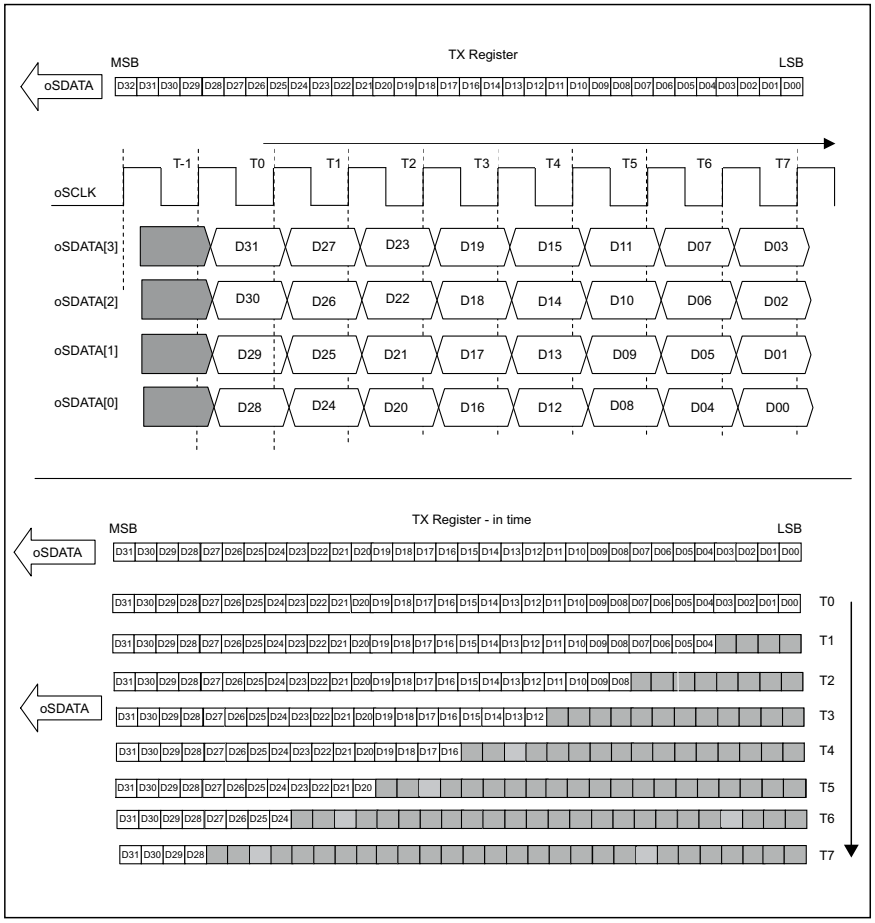
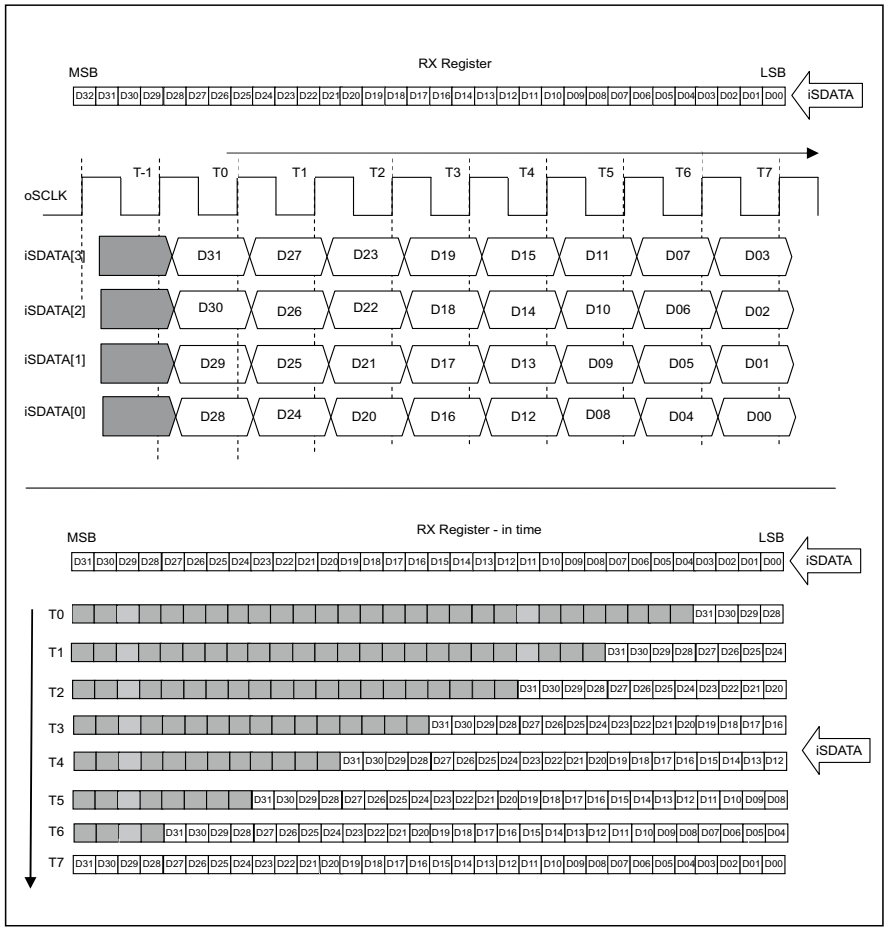


Figure 33-40. Shift direction for 32-bit registers - Quad mode RX (assumptions: HSSPIn_PCC0~3:CPOL = '0', HSSPIn_PCC0~3:CPHA = '0', HSSPIn_PCC0~3:SDIR = '0', HSSPIn_FIFOCFG:FWIDTH = '11')



33.3.6 Byte reordering

When SPI memory performs a read operation, Byte 0 is transmitted first followed by Byte 1, Byte 2 and Byte 3. If HSSPIn_PCC0~3:SDIR is set to ‘0’, then additional byte reordering is required for data to be correctly aligned on the AHB bus.

The following series of figures, Figure 33-41 to Figure 33-46 illustrates this reordering in Command Sequencer Mode and Direct Mode of operation. The series also shows that in both modes when HSSPIn_PCC0~3:SDIR is set to ‘0’, the most significant bit in the shift register is transmitted first and the data received first is shifted into the least significant bit in the shift register. When it is set to ‘1’, the least significant bit in the shift register is transmitted and the data received first is shifted into the most significant bit in the shift register.

Figure 33-41. Transmission in Command Sequencer Mode for quad bit SPI (HSSPIn_PCC0~3:SDIR = ‘0’)

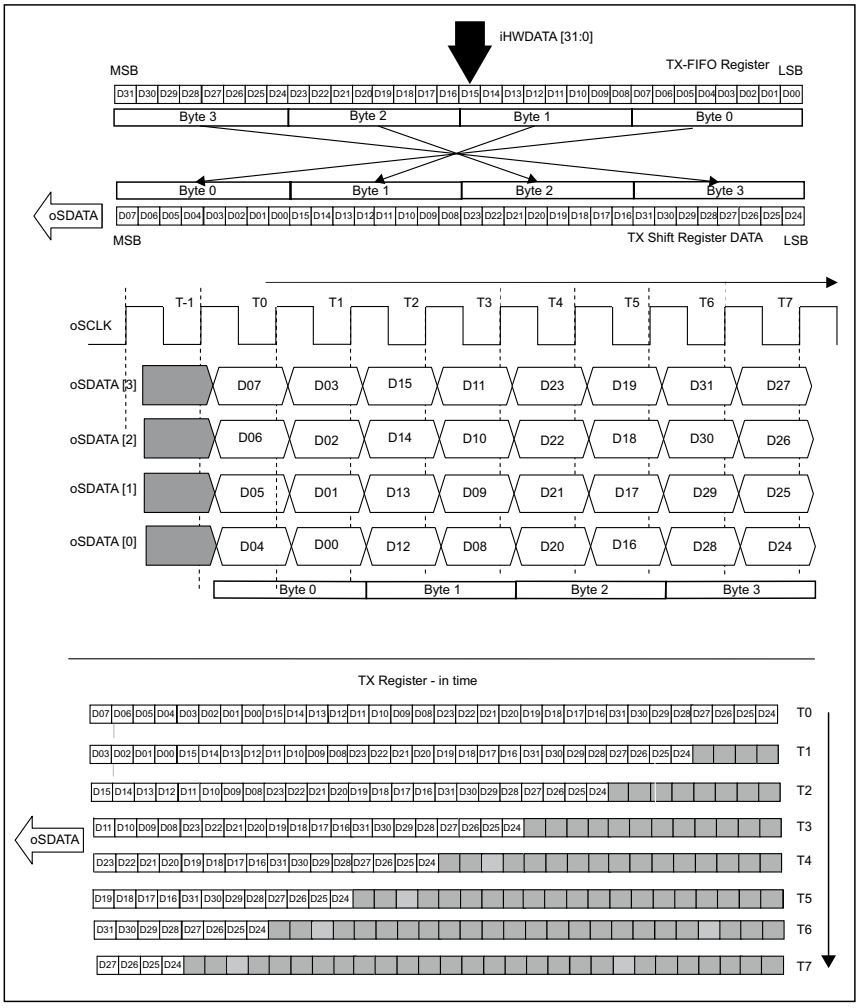


Figure 33-42. Reception in Command Sequencer Mode for quad bit SPI (HSSPIn_PCC0~3:SDIR = '0')

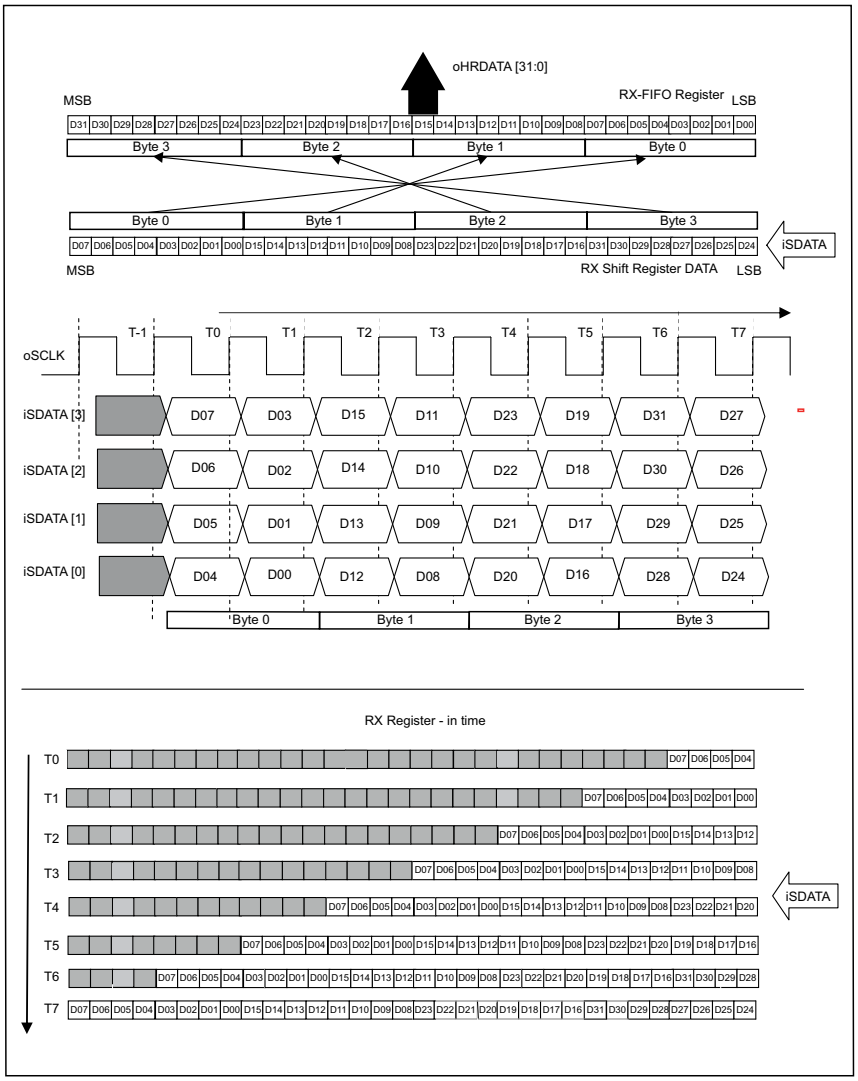
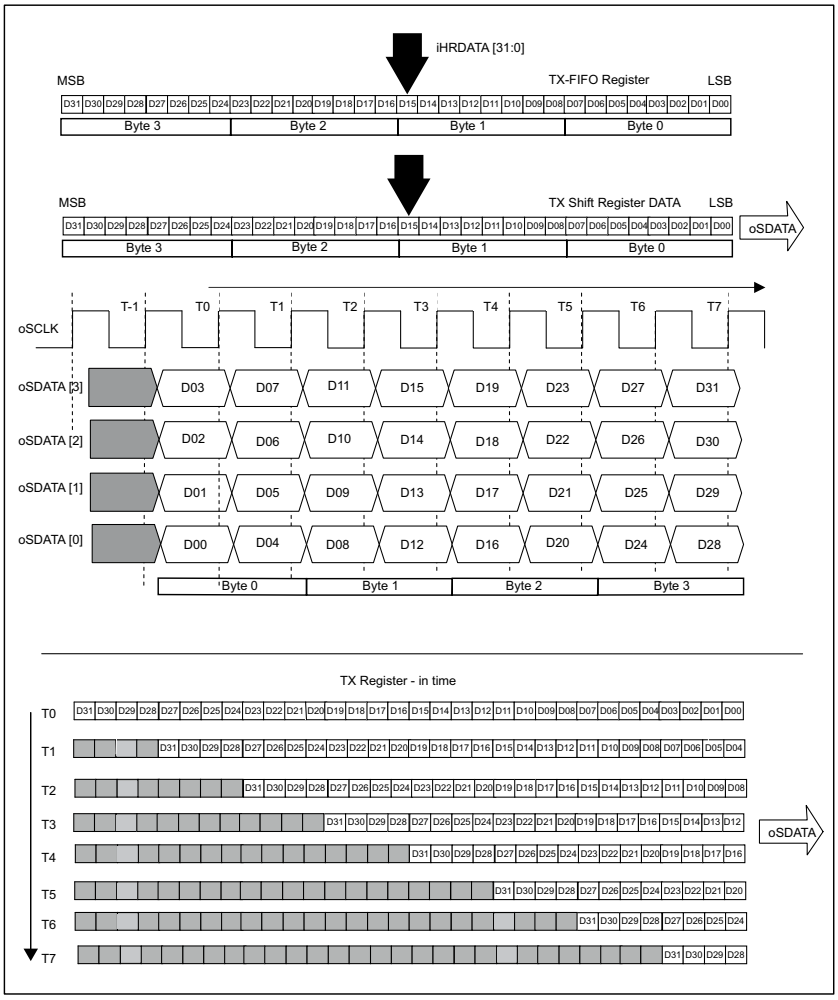
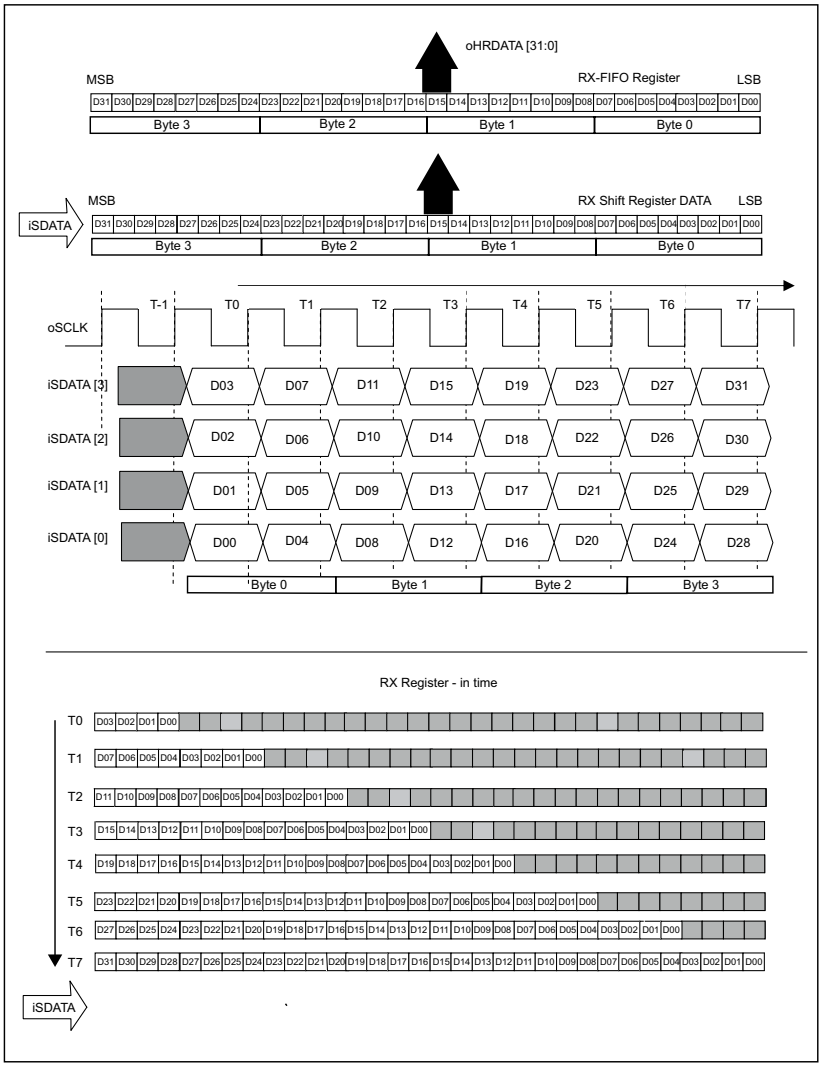


Figure 33-43. Transmission in Command Sequencer Mode for quad bit SPI (HSSPIn_PCC0~3:SDIR = '1')



Note: The waveforms above apply also to transmission in Direct Mode for quad bit SPI with HSSPIn_PCC0~3:SDIR = '1'

Figure 33-44. Reception in Command Sequencer Mode for quad bit SPI (HSSPIn_PCC0~3:SDIR = ‘1’)



Note: The waveforms above apply also to reception in Direct Mode for quad bit SPI with HSSPIn_PCC0~3:SDIR = ‘1’

Figure 33-45. Transmission in Direct Mode for quad bit SPI (HSSPIn_PCC0~3:SDIR = '0')

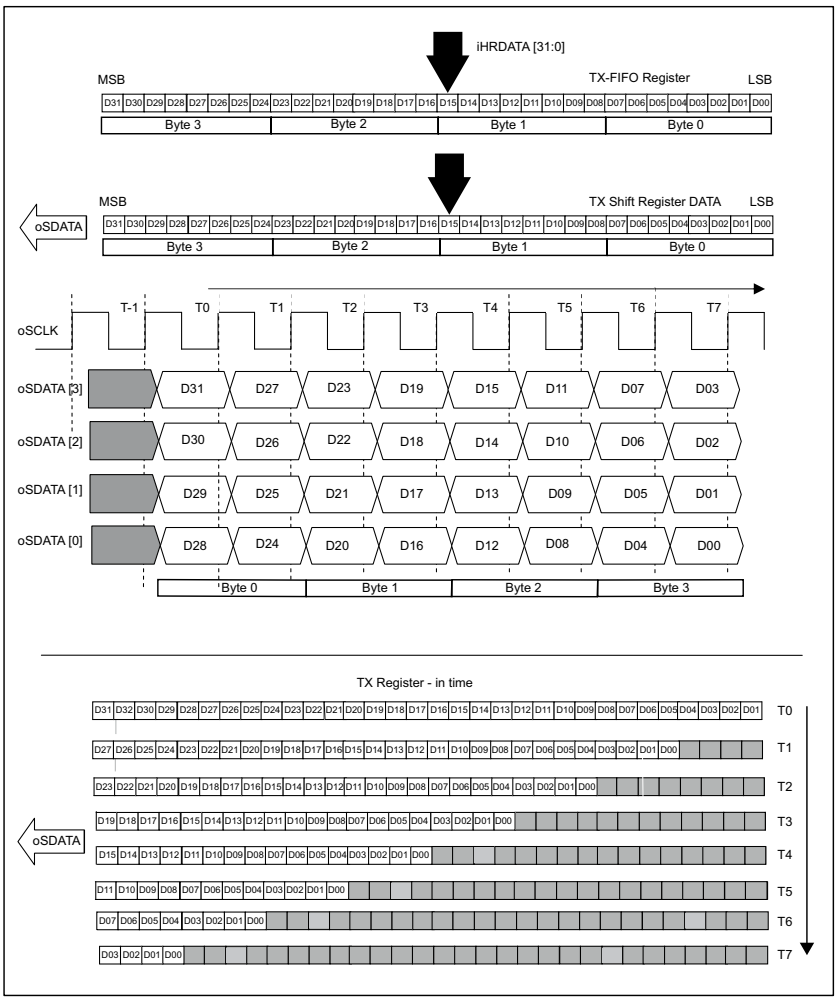
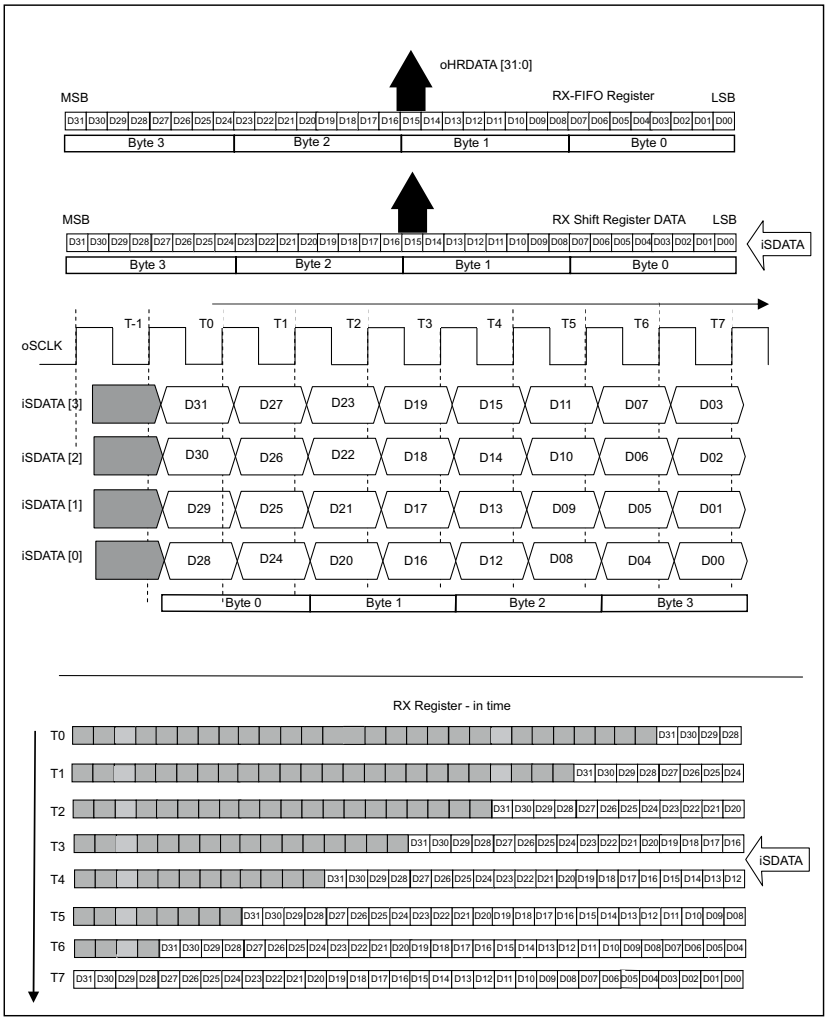


Figure 33-46. Reception in Direct Mode for quad bit SPI (HSSPIn_PCC0~3:SDIR = '0')



33.3.7 Safe synchronisation of internal data

While a serial transfer is in progress, HSSPI has to internally move the data across the two clock domains (AHB clock domain and the serial clock domain) using synchronizers which have their inherent latency.

Synchronisation in master mode

In certain cases when the synchroniser latency becomes a bottleneck in the serial transfer, the data will not be synchronised properly. To avoid this situation, the software programmers must ensure that the HSSPIn_PCC0~3:SAFESYNC is set in such cases.

The exact conditions when the setting of the HSSPIn_PCC0~3:SAFESYNC bit is required, depend on the transfer protocol, the width of the shift register and the ratio between the AHB clock frequency (i.e. Fhclk) and the Serial clock frequency (Fsclk).

When HSSPIn_PCC0~3:SAFESYNC bit is set, the HSSPI master halts the current serial transfer intermittently while it is internally synchronises the data. The duration of this halt is 3 periods of the serial clock. The serial interface is halted for the safe synchronisation of internal data only when all of the following three conditions are satisfied:

- The HSSPI is in master mode (i.e. HSSPIn_DMCFG:MST= '1' or HSSPIn_MCTRL:CSEN = '1')
- The HSSPIn_PCC0~3:SAFESYNC bit is set to '1'
- 'Width of shift register is 8 bits and serial interface is configured for dual bit or quad bit mode' or 'Width of shift register is 16 bits and serial interface is configured for quad bit mode'

Thus, when the HSSPIn_PCC0~3:SAFESYNC bit is set to '1' after the transfer of each data chunk, the SPI core on the fly decides whether or not to wait for safe synchronisation or not, depending on the width of the shift register that is being used at that particular instant. Merely setting the HSSPIn_PCC0~3:SAFESYNC bit does not imply that the SPI core would insert extra wait states for every chunk of data being transmitted. It inserts the extra wait states if and only if the specific conditions related to the width of the shift register and the SPI protocol (legacy/dual/quad) in use are satisfied. This ensures that the achievable bandwidth of the serial transfer is not severely impeded due to the time lost in safe synchronisation.

With reference to HSSPIn_PCC0~3:SAFESYNC, the term 'Width of shift register' used above is explained in the following sub-sections.

In direct mode, generally the width of the shift register is decided by the width of the FIFOs, configured in HSSPIn_FIFOCFG:FWIDTH. However, there are two special conditions that also need to be considered:

a. If the HSSPIn_FIFOCFG:TXCTRL bit in any of the TX-FIFO locations is set to '1', then the smallest width of the shift register used during the entire transfer is 1 byte. This width will be used by the HSSPI while deciding whether or not safe synchronisation is required for a transfer or not.
b. Specifically while using the counter mode (i.e. HSSPIn_DMBCC and HSSPIn_DMBCS registers) for stopping the serial transfer, if the number of bytes to be transferred (programmed in the HSSPIn_DMBCC register) is not divisible by the number of bytes configured in the FIFO width (i.e. HSSPIn_FIFOCFG:FWIDTH field), then the remainder of the division decides the smallest width of the shift register that is used during the transfer. Forexample, if HSSPIn_DMBCC:BCC is programmed with a value of 10 bytes and the HSSPIn_FIFOCFG:FWIDTH is programmed with a value of 4 bytes, then the HSSPI will perform the loading/unloading of the shift register in sets of 4 bytes, followed again by 4 bytes and the remaining 2 bytes is transferred before the transfer ends. Thus, in this case the smallest width of the shift register used during the entire transfer 2 bytes (assuming that the HSSPIn_FIFOCFG:TXCTRL bit in all TX-FIFO locations is 0). This width shall be used by the HSSPI while deciding whether safe synchronisation is required for a transfer.

Note: For all modes (legacy, dual and quad) which include reception, HSSPIn_DMBCC:BCC must be a multiple of the FIFO width (set in HSSPIn_FIFOCFG:FWIDTH). Table 33-34 lists the conditions when the HSSPIn_PCC0~3:SAFESYNC bit is set by the CPU to clock domain and the serial clock domain '1' while the HSSPI is configured in direct mode, master operation.

Table 33-34. Criteria for setting the HSSPIn_PCC0~3:SAFESYNC bit in direct mode, master operation

Mode	Width of shift register	Protocol	HSSPIn_PCC0~3: SAFESYNC shall be set to '1', if:	Maximum supported SCLK frequency
Direct mode, master	8 bits	Legacy	HSSPIn_PCC0~3:SAFESYNC is not required	Fsclk <= (1/2) Fhclk
		Dual bit	Fsclk <= (1/2) Fhclk	
		Quad bit	Fsclk <= (1/5) Fhclk	
	16 bits	Legacy	HSSPIn_PCC0~3: SAFESYNC is not required	
		Dual bit	HSSPIn_PCC0~3: SAFESYNC is not required	
		Quad bit	Fsclk <= (1/2) Fhclk	
	24 bits	Legacy	HSSPIn_PCC0~3: SAFESYNC is not required	
		Dual bit		
		Quad bit		
	32 bits	Legacy	HSSPIn_PCC0~3: SAFESYNC is not required	
		Dual bit		
		Quad bit		

In command sequencer mode while the command sequence is being transmitted, the width of the shift register is 8 bits, and while the data is being written/read (to/from the serial memory), the width of the shift register is equal to the AHB bus transfer size.

Table 33-35 lists the conditions where setting the HSSPIn_PCC0~3:SAFESYNC bit to '1' is required while operating in command sequencer mode.

Table 33-35. Criteria for setting the HSSPIn_PCC0~3:SAFESYNC bit in command sequencer mode

Mode of operation	AHB transfer size of the memory-mapped transfer	Protocol	HSSPIn_PCC0~3: SAFESYNC shall be set to '1' if:	Maximum supported SCLK frequency
Command sequencer	8 bits	Legacy	HSSPIn_PCC0~3: SAFESYNC is not required	Fsclk <=(1/2) Fhclk
		Dual bit	Fsclk <= (1/2) Fhclk	
		Quad bit	Fsclk <= (1/6) Fhclk	Fsclk <= (1/2) Fhclk
	16 bits	Legacy	HSSPIn_PCC0~3: SAFESYNC is not required	Fsclk <=(1/2) Fhclk
		Dual bit	Fsclk <= (1/2) Fhclk	
		Quad bit	Fsclk <= (1/5) Fhclk	
	32 bits	Legacy	HSSPIn_PCC0~3: SAFESYNC is not required	
		Dual bit	Fsclk <= (1/2) Fhclk	
		Quad bit	Fsclk <= (1/5) Fhclk	

Synchronisation in slave mode

While the HSSPI is configured in slave mode, the external SPI master must control the flow of the serial data to avoid the loss of synchronisation.

Refer to Table 33-36. It lists the maximum supported Fsclk values.If the serial clock frequency is more than the value listed in this Table 33-36, while operating in slave mode then data would be unpredictable. Therefore, in slave mode, the SPI master must be configured to generate serial clock frequencies that are less than the maximum value of Fsclk shown in Table 33-36.

Table 33-36. Maximum supported SCLK frequency in slave mode

Mode	Width of shift register	Protocol	HSSPIn_PCC0~3: SAFESYNC value	Maximum supported SCLK Frequency
Direct mode, slave	8 bits	Legacy	Safe synchronisation is not applicable in slave mode.	Fsclk <= (1/2) Fhclk
		Dual bit		Fsclk <= (1/2) Fhclk
		Quad bit		Fsclk <= (1/2) Fhclk
	16 bits	Legacy		Fsclk <= (1/2) Fhclk
		Dual bit		Fsclk <= (1/2) Fhclk
		Quad bit		Fsclk <= (1/2) Fhclk
	24 bits	Legacy		Fsclk <= (1/2) Fhclk
		Dual bit		
		Quad bit		
	32 bits	Legacy		
		Dual bit		
		Quad bit		

33.3.8 Debug mode

The HSSPI can be configured in debug mode using the HSSPIn_MCTRL:DBGEN bit.

If the iDEBUG input signal of the HSSPI is asserted when the HSSPI is in debug mode and is working as a master in direct mode of operation, the serial interface is halted by stopping the activity on the serial clock output of HSSPI. The clock is stopped as long as the iDEBUG input is asserted and the debug mode is enabled. While the serial interface is halted, the CSR registers of the HSSPI can still be accessed normally by the AHB master.

The activity on the serial clock resumes either when the iDEBUG input is de-asserted or when the debug mode is disabled in the HSSPIn_MCTRL:DBGEN bit.

33.4 Direct mode

In direct mode the MCU (or the DMA engine) is responsible for directly controlling the serial transfer on the serial interface. The Direct mode of transfer can be enabled using the HSSPIn_MCTRL:CSEN bit. In direct mode, the HSSPI uses its internal FIFOs to temporarily store the data that is transmitted and received over the serial interface. This section describes the direct mode of operation.

Internal FIFOs

Internally, the HSSPI has two FIFOs for temporary storage: one for the data to be transmitted and the other for data received.

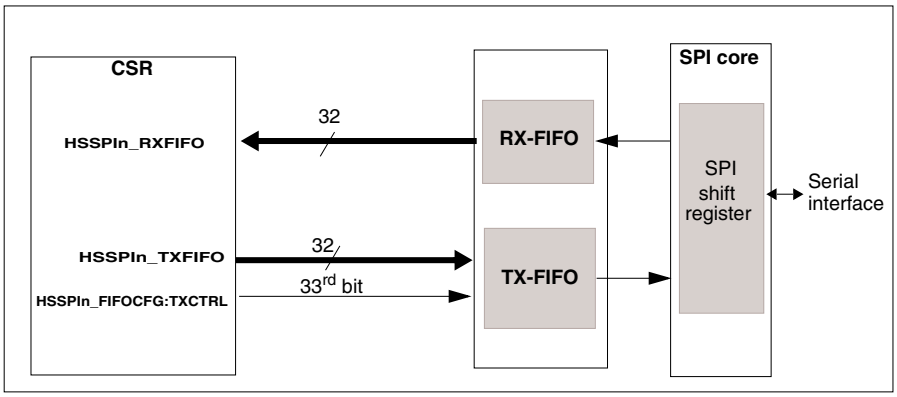
Based on whether the serial transfers in the HSSPI are configured as TX only, RX only or TX and RX in the HSSPIn_DMTRP:TRP field, only one or both FIFOs are used by the HSSPI. If the HSSPI is configured for TX-only operation, the TX-FIFO is used. If HSSPI is configured for RX only operation, the RX-FIFO is used. If it is configured for TX and RX operation, both FIFOs are used.

FIFO size

Each FIFO is 16 locations deep and has a data width of 32 bits. The software can configure the valid data width of the TX-FIFO and the RX-FIFO in HSSPIn_FIFOCFG:FWIDTH.

The shift register in the SPI core is 32-bits wide. When the width of the FIFO is changed in the HSSPIn_FIFOCFG:FWIDTH field, the usable width of the shift register also changes accordingly. For details refer to Figure 33-47.

Figure 33-47. HSSPI in direct mode



In addition to a 32-bit data width, each location in TX-FIFO has a 33rd control bit (HSSPIn_FIFOCFG:TXCTRL), which decides whether the data from the TX-FIFO is transmitted by the SPI core or whether the serial data lines are tri-stated. If the HSSPIn_FIFOCFG:TXCTRL bit is set to '1', the HSSPI further decodes the bit 0 of the data in the corresponding TX-FIFO location. All possibile combinations for the HSSPIn_FIFOCFG:TXCTRL bit and the bit 0 of the TX-FIFO data are listed in Table 33-37.

Table 33-37. Tri-stating the serial data output lines during transmission

HSSPIn_FIFOCFG: TXCTRL	Bit 0 of TX-FIFO data	Description
'0'	Don’t care	Serial data output lines are not tri-stated while transmitting the corresponding data.
'1'	'0'	Serial data output lines are tri-stated for 1 byte time. The data from the corresponding TX-FIFO location is not transmitted.

Table 33-37. Tri-stating the serial data output lines during transmission

HSSPIn_FIFOCFG: TXCTRL	Bit 0 of TX-FIFO data	Description
'1'	'1'	Irrespective of the shift direction configured in HSSPIn_PCC0~3:SDIR bits, the data transmission takes place in the following order: 1. Data bits [7:4] from the corresponding TX-FIFO location are transmitted. Direction of transmission of this data depends on configuration of HSSPIn_PCC0~3:SDIR bit. 2. SDATA output lines are tri-stated for 4 bit times.

For all calculations (e.g. while using the HSSPIn_DMBCC:BCC feature or while referring to 31.3-2), the data in the TX-FIFO location for which the HSSPIn_FIFOCFG:TXCTRL bit is set, is considered to be 1 byte wide. The HSSPIn_FIFOCFG:TXCTRL bit associated with the data words in the TX- FIFO allows the software to generate the command sequences (using direct mode) for interfacing with some external quad-bit SPI memories, which need to tri-state the serial data (i.e. SDATA) lines during the 'dummy' cycles or during the transmission of lower nibble of the ‘mode bits’ of the command sequence.

FIFO accesses

Irrespective of the configured width of the FIFOs, only 32-bit word accesses are allowed to the HSSPIn_RXFIFO and HSSPIn_TXFIFO registers.

A read access to the HSSPIn_RXFIFO register in the CSR directly pops out a word from the RX- FIFO. If the RX-FIFO width was set to 8 bits, the most-significant 24 bits read out from HSSPIn_RXFIFO register are logic 0. Similarly, when the FIFO width is set to 16 bits or 24 bits, the unused bits read out from the HSSPIn_RXFIFO register are logic 0.

A write access to the HSSPIn_TXFIFO register in the CSR pushes a word of data and a HSSPIn_FIFOCFG:TXCTRL bit (see HSSPIn_FIFOCFG:TXCTRL) into the TX-FIFO. However, when the HSSPI transmits the data on the serial lines, it uses only the least significant bits of the data read from the HSSPIn_TXFIFO register. The number of these least significant bits that are transmitted depends on the configured width of the TX-FIFO. The unused most significant bits are ignored by the HSSPI.

Accessing the RX data

The serial data received by the HSSPI on the SDATA lines are assembled in a shift register (in the SPI core) before being pushed into the RX-FIFO.

When a transfer completes (i.e. slave select line is de-asserted), the HSSPIn_RXSHIFT register is updated by with the assembled data and the HSSPIn_RXBITCNT:RXBITCNT field is updated with the number of valid bits in the HSSPIn_RXSHIFT register. When the HSSPIn_TXF:TSSRS or the HSSPIn_RXF:RSSRS interrupt flag is set, the software can read the HSSPIn_RXSHIFT and HSSPIn_RXBITCNT:RXBITCNT field to get the RX data which has not yet been pushed into the RX-FIFO.

Service requests

When operating in direct mode, the triggering of the interrupt service request to the MCU is based on the current fill levels of the TX-FIFO and the RX-FIFO and their configured threshold values. Alternatively, the external DMA engine can be used for data transfers. The HSSPI has an interface with the DMA engine in the MCU for transfers to/from its TX-FIFO and the RX-FIFO when operating in direct mode. Interrupt flags are also set when the current SPI transfer finishes.

Assertion of interrupt service requests based on FIFO levels

The fill levels of both FIFOs are accessible to the system through the HSSPIn_DMSTATUS:TXFLEVEL and the HSSPIn_DMSTATUS:RXFLEVEL fields. The interrupt service requests generated by the HSSPI are based on the FIFO fill levels and their threshold values, which are configured by the MCU.

The 'TX-FIFO fill level less than or equal to threshold' (i.e. HSSPIn_TXF:TFLETS) interrupt flag is set if the TX-FIFO fill level (i.e HSSPIn_DMSTATUS:TXFLEVEL) is less than or equal to the TX-FIFO threshold value configured in HSSPIn_FIFOCFG:TXFTH.

The 'RX-FIFO fill level more than threshold' (i.e. HSSPIn_RXF:RFMTS) interrupt flag is set if the RX-FIFO fill level (i.e. HSSPIn_DMSTATUS:RXFLEVEL) is more than the RX-FIFO threshold value configured in HSSPIn_FIFOCFG:RXFTH.

If the HSSPI is configured for TX only operation, the RX-FIFO is not used. If the HSSPI is configured for RX only operation, the TX-FIFO is not used.

Assertion of DMA service requests based on FIFO levels

The HSSPI supports the block transfer mechanism of the DMA engine. The DMA service requests are generated by the HSSPI and are based on the FIFO fill levels and their threshold values, which are configured by the MCU. To keep track of the number of successful data transfers to/from the TX-FIFO and/or the RX-FIFO, the HSSPI internally maintains two down counters: HSSPI RX block counter and HSSPI TX block counter. Each of these counters is a 5-bit down counter, which is reloaded with the DMA block size (for the respective channel) whenever the DMA service request for that channel is asserted. The counters are decremented with every successful read (or write) access to the RX-FIFO (or TX-FIFO). In the case of RX-FIFO accesses, the RX block counter is decremented only if the access was from an AHB master other than the DAP controller. The block counters do not underflow (i.e. the counter value remains ‘0’ even if it tried to decrement while it was already '0').

Each DMA read and write channel has a dedicated DMA block-size fault-status flag in the HSSPIn_FAULTF register. A DMA block-size fault is triggered if all of the following conditions are satisfied:

1. The DMA block counter is decremented (due to a valid AHB access) while it is already '0'.
2. The DMA enable bit (HSSPIn_DMDMAEN:RXDMAEN or HSSPIn_DMDMAEN:TXDMAEN) for the corresponding DMA channel is set to '1'.
3. The Module is enabled (i.e. HSSPIn_MCTRL:MES = '1').
4. The Module is operating in direct mode of operation (i.e. HSSPIn_MCTRL:CSEN = '0').

The DMA read channel must be setup to perform a block transfer of 'HSSPIn_FIFOCFG:RXFTH + 1' transfers. The DMA write channel must be setup to perform a block transfer of '16 - HSSPIn_FIFOCFG:TXFTH' transfers. These values are reloaded into the HSSPI module’s internal block counters whenever the DMA read/write channel service request is asserted.

The DMA block counter is reset to '0' by one or more of the following conditions:

1. The corresponding DMA channel is disabled (in HSSPI_DMDMAEN register).
2. The Module is completely disabled (i.e. HSSPIn_MCTRL:MES = '0').
3. The Mode of operation is switched from direct mode to command sequencer mode.HSSPI

Note: If the DMA Block size is configured incorrectly, it can happen that an error response is received later than expected.

The RX DMA service request (for the DMA read channel) is asserted if all of the following conditions are satisfied:

1. The RX-FIFO fill level (i.e. HSSPIn_DMSTATUS:RXFLEVEL) is more than the RX-FIFO threshold value configured in HSSPIn_FIFOCFG:RXFTH. This condition is the same as the HSSPIn_RXF:RFMTS bit.
2. The HSSPI RX block counter value is '0'.
3. The DMA read channel acknowledgement signal is deasserted by the DMA engine.
4. The Previous service request for DMA read channel is not pending.
5. The DMA read channel service request is enabled in the HSSPIn_DMDMAEN:RXDMAEN bit, AND
6. The DMA read channel block-size fault-interrupt flag is not set (i.e. HSSPIn_FAULTF:DRCBFES = '0').
7. The Module is enabled (i.e. HSSPIn_MCTRL:MES = '1').
8. The Module is in direct mode (i.e. HSSPIn_MCTRL:CSEN = '0').
9. The configured transfer protocol (in HSSPIn_DMTRP) is such that RX-FIFO is used. e.g. if HSSPI is configured for TX only operation, the RX-FIFO is not used and DMA read service request is not asserted in such cases.

The RX DMA service request (for DMA read channel) is deasserted if one or more of the following conditions is satisfied:

1. The DMA read channel service request has been acknowledged by the DMA engine.
2. DMA read channel service requests have been disabled (i.e. HSSPIn_DMDMAEN:RXDMAEN = '0').
3. The module is completely disabled (i.e. HSSPIn_MCTRL:MES = '0').
4. The Mode of operation is switched from direct mode to command sequencer mode.

The TX DMA service request (for DMA write channel) is asserted if all of the following conditions are satisfied:

1. The TX-FIFO fill level (i.e. HSSPIn_DMSTATUS:TXFLEVEL) is less than or equal to the threshold value configured in HSSPIn_FIFOCFG:TXFTH. This condition is the same as the HSSPIn_TXF:TFLETS interrupt flag.

2. The HSSPI TX block counter value is '0'.
3. The DMA write channel acknowledgement signal is de-asserted by the DMA engine.
4. The Previous service request for the DMA write channel is not pending.
5. The DMA write channel service request is enabled in the HSSPIn_DMDMAEN:TXDMAEN bit.
6. The DMA write channel block size fault interrupt flag is not set (i.e. HSSPIn_FAULTF:DWCBSFS = '0').
7. The Module is enabled (i.e. HSSPIn_MCTRL:MES = '1').
8. The Module is in direct mode (i.e. HSSPIn_MCTRL:CSEN = '0').
9. The configured transfer protocol (in HSSPIn_DMTRP) is such that TX-FIFO is used. E.g. if the HSSPI is configured for RX only operation, the TX-FIFO is not used and the DMA write service request is not asserted in such cases.

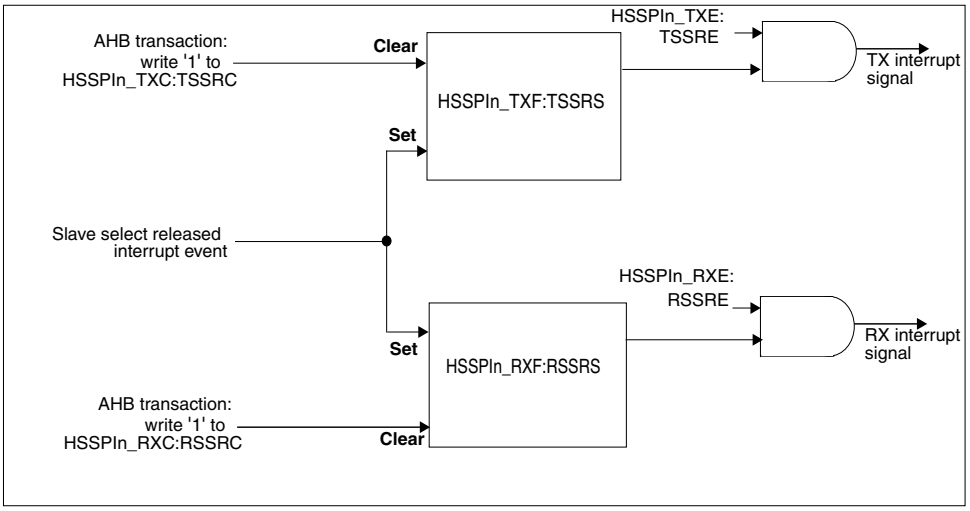
The TX DMA service request (for DMA write channel) is de-asserted if one or more of the following conditions are satisfied:

1. A DMA write channel service request has been acknowledged by the DMA engine.
2. DMA write channel service requests have been disabled (i.e. HSSPIn_DMDMAEN:TXDMAEN = '0').
3. The Module is completely disabled (i.e. HSSPIn_MCTRL:MES = '0').
4. The Mode of operation is switched from direct mode to command sequencer mode.

Assertion of service requests on end of transfer

In direct mode, the HSSPI also triggers interrupts when the slave select line is de-asserted. The slave select de-assertion event is routed onto two interrupt flags: HSSPIn_TXF:TSSRS and HSSPIn_RXF:RSSRS, which have separate interrupt clear and interrupt enable bits. The interrupt flags are routed onto separate interrupt signals. This is shown in Figure 33-48.

Figure 33-48. Routing of the 'slave select released' interrupt event



SPI transfers in master mode

When the HSSPI is the SPI master, it initiates the transfers onto one of the four SPI slave select lines, selected by the HSSPIn_DMPSEL:PSEL field.

Communication attributes of the HSSPI in master mode

Communication over the serial interface has several attributes such as frequency, polarity and phase of the serial interface clock, polarity of the slave select line, etc. These communication attributes may be different for several external connected SPI devices. When the HSSPI is working as a master in direct mode of operation, it can be interfaced with up to four slaves, each with different attribute values.

These device-specific communication attributes can be configured in the HSSPIn_PCC0~3 registers in CSR.

Initiating the serial transfers

When the HSSPI is enabled (i.e. HSSPIn_MCTRL:MEN = '1') and working as a master (i.e. HSSPIn_DMCFG:MST = '1') in direct mode (i.e. HSSPIn_MCTRL:CSEN = '0'), serial transfers are initiated by the HSSPI when the HSSPIn_DMSTART:START bit is set to '1'.

If the HSSPIn_DMTRP:TRP is programmed such that transmission is enabled, and if the TX-FIFO is empty when the HSSPIn_DMSTART:START bit is set to '1', then the HSSPI delays the initiation of the serial transfer until the TX-FIFO is written to the software.

While HSSPI has delayed the initiation of the serial transfer (until TX-FIFO is written by software):

a. If the HSSPIn_MCTRL:MEN bit is reset to '0' by software, then the disabling of the module will take precedence over starting the next transfer.
b. If counter mode is used for controlling the transfer length, then the HSSPIn_DMBCS register will be loaded with the value in HSSPIn_DMBCC register only when the serial transfer is initiated by the HSSPI. Until then, the HSSPIn_DMBCS register will maintain its '0' value.

Note:

Once the HSSPIn_DMSTART:START bit is set to '1', it cannot be reset by the software anymore. The HSSPI module resets the bit after it has started the serial transfer. Writing a '1' to the HSSPIn_DMSTART:START bit while it is already set to '1' has no effect. Writing a '1' to the HSSPIn_DMSTART:START bit while it is '0' and a serial transaction is already in progress does not affect the ongoing transfer. A new serial transfer is initiated after the current transfer completes.

Halting a transfer due to lack of TX DATA or due to lack of RX-FIFO space

As per the standard SPI protocol, an ongoing transfer can be halted by keeping the slave select asserted and by cutting the serial clock. The HSSPI automatically cuts the serial clock while it is waiting for the TX-FIFO to be written to or while it is waiting for the RX-FIFO to be read.

Depending on whether the HSSPI master is operating as TX only, RX only or TX and RX, there are three scenarios in which the HSSPI cuts the serial clock and halts a transfer:

TX only mode	The serial clock is cut when the TX-FIFO and shift register are empty.
RX only mode	The serial clock is cut when the RX-FIFO and shift register are full.
TX and RX mode	The serial clock is cut when a) The TX-FIFO and the shift registers are empty. b) RX-FIFO and the shift registers are full.

When an ongoing transfer is halted (by cutting the serial clock) by the HSSPI due to the unavailability of FIFO resources, the corresponding slave-select line is kept asserted, indicating to the slave that the transfer has not been finished. The halted transfers are automatically resumed by the HSSPI (by starting toggling of the serial clock) when the FIFO resources become available.

Controlling the transfer length

In master mode the transfer length (i.e the de-assertion of the slave-select lines) can be controlled in two ways:

- Counter mode and
- Software flow control mode

These modes can be selected through the HSSPIn_DMCFG:SSDC bit.

In counter mode the MCU is supposed to initialize the HSSPIn_DMBCC:BCC field with the number of bytes to be transferred over the serial interface before the slave-select (i.e. SSEL) output is deasserted. When the HSSPI transfers are initiated, the HSSPI counts the number of bytes that are transferred and releases the slave-select signal after the number of bytes indicated in HSSPIn_DMBCC:BCC have been transferred.

In software flow control mode the MCU controls the transfer length by using the HSSPIn_DMSTOP:STOP bit. Depending on whether the HSSPI master is operating as TX only, RX only or TX and RX, the de-assertion of slave-select output is controlled in the following ways:

TX only mode	The transfer is completed when the HSSPIn_DMSTOP:STOP bit is set and all contents of TX-FIFO are transmitted

RX only mode	The transfer is completed when the HSSPIn_DMSTOP:STOP bit is set and all bits in the shift register (used in the SPI core for assembling the received serial data) are shifted in.
TX and RX mode	The transfer is completed when the HSSPIn_DMSTOP:STOP bit is set and all the contents of TX-FIFO are transmitted.

SPI transfers in slave mode

When the HSSPI is configured as a slave (i.e. HSSPIn_DMCFG:MST = '0'), it responds to the transfers initiated by the external masters when its slave-select input (i.e. SSEL0) is asserted.

Communication attributes of the HSSPI in slave mode

Communication over the serial interface has several attributes such as polarity and phase of the serial interface clock, polarity of the slave select line, etc. These communication attributes may be different for several external connected SPI devices. When the HSSPI is working as a slave in direct mode of operation, its communication attributes can be configured in the HSSPIn_PCC0 register in the CSR.

While in slave mode, the serial clock (SCLK) is driven by the external master.

Lack of FIFO resources while a serial transfer is ongoing

When the HSSPI acts as a slave, it has no control over the serial transfer. Therefore, it is the responsibility of the software to monitor the HSSPIn_TXF:TFLETS, HSSPIn_TXF:TFMTS, HSSPIn_RXF:RFLETS and the HSSPIn_RXF:RFMTS interrupt flags and ensure the availability of FIFO resources.

However, in slave mode and while the HSSPI is configured for transmission (i.e. TX only, or TX and RX configuration in the HSSPIn_DMTRP:TRP field), the TX-FIFO never becomes empty (It should contain dummy data). Therefore, the HSSPI never sets the HSSPIn_TXF:TFES interrupt flag.

The data transmission continues with the data it gets from the TX-FIFO on the serial data lines, as long as the slave select is asserted and the clock is toggling.

While the HSSPI is configured for reception (i.e. RX only or TX and RX configuration in HSSPIn_DMTRP:TRP field), if the RX-FIFO becomes full, the HSSPI sets the HSSPIn_RXF:RFFS interrupt flag and keeps overwriting the serial data received on the SDATA lines into the RX-FIFO.

33.5 Command sequencer mode

In command sequencer mode the HSSPI acts as an SPI master for interfacing with the external serial memory devices. Each of the four slave select lines can be used for mapping uniform types of 'serial flash' or 'serial SRAM' devices. Memory accesses initiated by the MCU and other AHB masters on the AHB bus are automatically converted to serial memory read/write commands by the HSSPI. This section describes the command sequencer mode of operation of the HSSPI.

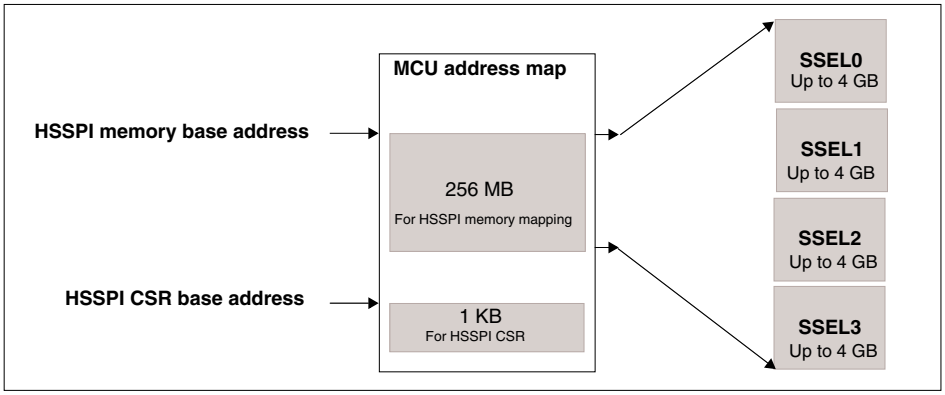
Memory mapping

The command sequencer mode can be used for memory mapping up to four serial Flash or serial SRAM memory devices on the four slave select outputs of the HSSPI. All memory mapped devices should be of the same family.

In command sequencer mode, the HSSPI is allocated a memory space of 256 MBs for mapping up to four external memory devices. Each slave select can theoretically address a memory of up to 4 GB (i.e. 32-bit address bus) by using the address extension mechanism in the command sequencer. The address extension mechanism allows the concatenation of the most significant bits from a 19-bit Address Extension Register (i.e. HSSPIn_CSAEXT register) with the few bits from the AHB address bus to form a 32-bit address to be accessed on each slave select. This feature is explained in detail in the subsequent sub sections of this chapter.

Thus, the 256 MB of MCU address space is virtually mapped to 16 GB of external serial memory, as shown in Figure 33-49.

Figure 33-49. Mapping of memory devices on the slave select lines



Selection of slaves

The HSSPIn_CSCFG:MSEL field indicates the size of the AHB address space associated with each slave select line. Based on the value of the HSSPIn_CSCFG:MSEL field and the address placed by the MCU (or another AHB master, such as the DMA controller) on the AHB address bus, the HSSPI command sequencer decides which of the four slave select lines is asserted. For more details refer to Table 33-38.

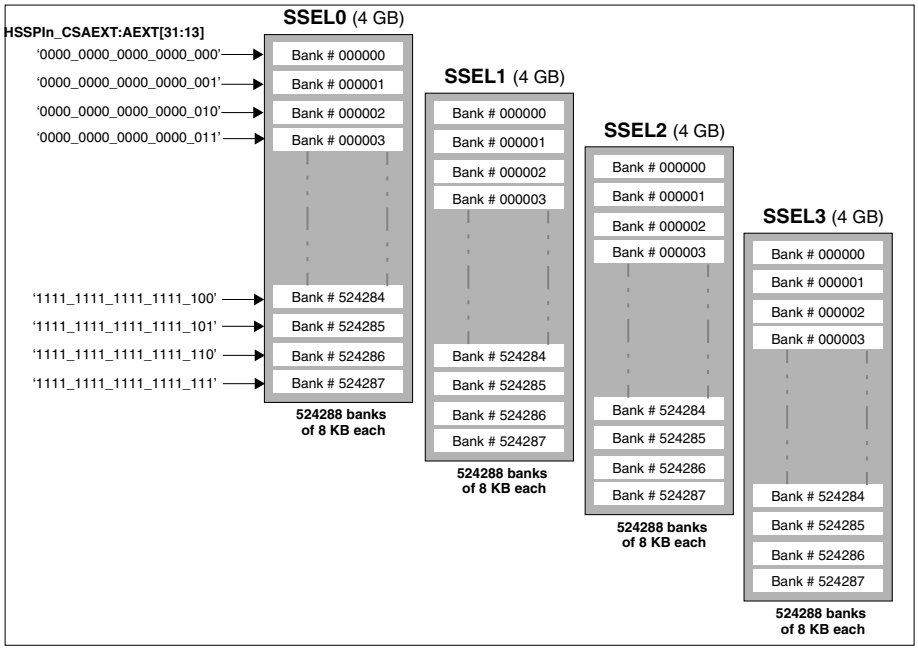
Assume that HSSPIn_CSCFG:MSEL indicates that the AHB address space associated with each slave select is 8 KB. If the AHB address is between 'HSSPI memory base address' and 'HSSPI memory base address + 8 KB', then slave select 0 is asserted. If the AHB address is between 'HSSPI memory base address + 8 KB' and 'HSSPI memory base address + 16 KB', then slave select 1 is asserted, and so on. If the AHB address is beyond 'HSSPI memory base address + 32 KB', then the address is out of range and HSSPIn_FAULTF:UMAFS interrupt flag is set.

Generation of 32-bit memory address

The mapping of 256 MB of MCU address space to 4 GB of address space on a slave select line is possible due to the address extension mechanism. Every memory device can be viewed as consisting of several memory banks. The size of each bank can be programmed in the HSSPIn_CSCFG:MSEL field, and a particular bank in a selected slave is chosen by changing the value in the HSSPIn_CSAEXT register. This method of reprogramming the HSSPUn_CASAEXT resgister each time a new bank in a selected slave is to be accessed allows a memory device up to 4GB to be addressed.

Figure 33-50 shows how each 4 GB device consists of 524288 banks when the HSSPIn_CSCFG:MSEL field is programmed to '0000'.

Figure 33-50. Addressing 4 GB devices on each slave select through different banks (HSSPIn_CSCFG:MSEL = ’0000’)



The least significant bits of the AHB address received by HSSPI on the AHB bus are used as the offset within the bank selected by the address extension bits. The concatenation of the appropriate number of bits from the Address Extension Register with the appropriate number of bits from the AHB address bus gives the 32-bit address of the memory to be accessed on the serial interface. Refer to Table 33-38

Table 33-38. MCU address space to memory address mapping

HSSPIn_CSCFG: MSEL field	Size of a memory bank on each slave select/size of the AHB address range associated with each slave select	Number of slave select lines that can be activated	Number of bits used from HSSPIn_CSA EXT register, for selection of the memory bank on a slave select	Number of bits used from AHB address bus for addressing the memory location within a bank
'0000'	8 KB	SSEL0, SSEL1, SSEL2, and SSEL3	AEXT[31:13]	HADDR[12:0]
'0001'	16 KB		AEXT[31:14]	HADDR[13:0]
'0010'	32 KB		AEXT[31:15]	HADDR[14:0]
'0011'	64 KB		AEXT[31:16]	HADDR[15:0]
'0100'	128 KB		AEXT[31:17]	HADDR[16:0]
'0101'	256 KB		AEXT[31:18]	HADDR[17:0]
'0110'	512 KB		AEXT[31:19]	HADDR[18:0]
'0111'	1MB		AEXT[31:20]	HADDR[19:0]
'1000'	2 MB		AEXT[31:21]	HADDR[20:0]
'1001'	4 MB		AEXT[31:22]	HADDR[21:0]
'1010'	8 MB		AEXT[31:23]	HADDR[22:0]
'1011'	16 MB		AEXT[31:24]	HADDR[23:0]
'1100'	32 MB		AEXT[31:25]	HADDR[24:0]
'1101'	64 MB		AEXT[31:26]	HADDR[25:0]
'1110'	128 MB	SSEL0 and SSEL1 only	AEXT[31:27]	HADDR[26:0]
'1111'	256 MB	SSEL0 only	AEXT[31:28]	HADDR[27:0]

Last two columns in Table 33-38 indicate which bits from HSSPIn_CSAEXT:AEXT and the AHB address bus (i.e. HADDR) are concatenated to get the final 32-bit address of the serial memory.

Although the final memory address generated in this way is a 32-bit address, it must be noted that the software can choose the number of bytes (from this 32-bit address) to send to the serial memory device during the address phase of a memory read/write command sequence.

Initiation of command sequence

Whenever the command sequencer receives an AHB read access for the memory mapped serial device, it initiates a corresponding memory read command on one of the four slave select lines, assembles the data it has received and responds with the memory read data.

Similarly, if the sequencer receives an AHB write access for the memory mapped serial device, it initiates a corresponding memory write command on one of the four slave select lines and transmits the data to be written serially on the SDATA lines.

While the HSSPI initiates a memory read command and receives the read data from the serial memory device, the AHB slave port of the HSSPI inserts wait states on the AHB bus. Similarly, wait states are also inserted for serial memory write sequences.

The HSSPI keeps track of the previous address and the AHB transfer type issued by the AHB master. If the new transaction address is not contiguous or if there is a switch in the command (from read to write or vice-versa), then a new command is issued on the serial interface.

AHB idle timeout

After a serial device is accessed in the command sequencer mode, the HSSPI keeps asserting the slave select line even when the serial transaction is over. Within the time period defined by The HSSPIn_CSITIME:ITIME field, if a new AHB transaction is detected on the AHB and the following conditons occur :

1. Address is contiguous with the previous transaction
2. Access type the same as previous command (i.e. read/write), then the HSSPI continues with the same serial transfer instead of initiating a new command sequence phase thereby, reducing the access time.

If there are subsequent AHB accesses to a non-consecutive memory address during the idle time or if the command (i.e. read/write) changes, then even before the idle-timer expires, the HSSPI deasserts the slave select (indicating the termination of the current transfer) and initiates a fresh transfer. If there are no subsequent AHB accesses to the consecutive memory address during the idle time, then after the idle timer expires the HSSPI de-asserts the slave select, indicating the termination of the transfer.

Thus, the HSSPIn_CSITIME:ITIME field is used to enhance the overall performance of the memory-mapped accesses by continuing the previous serial transaction for a configurable period. The unit of the time period defined by HSSPIn_CSITIME:ITIME field is the time period of the AHB clock input.

Configuration of command sequence in the CSR

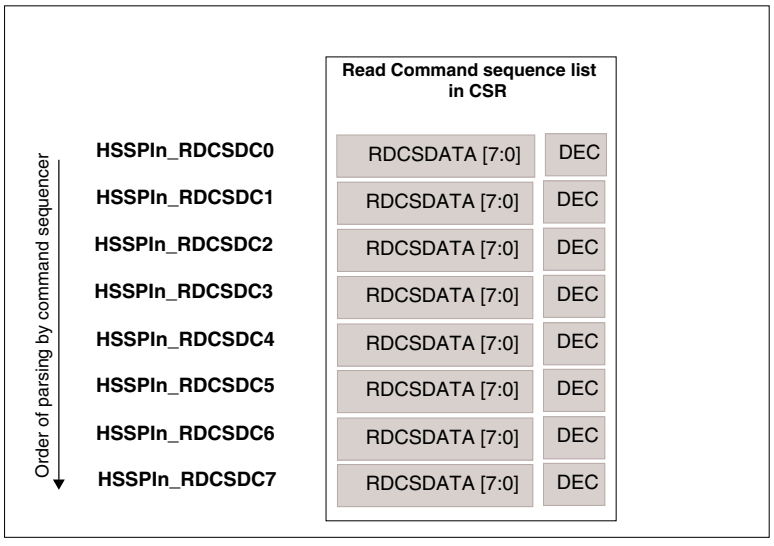
The Command sequencer supports memory read accesses. Optionally, if serial SRAM devices are memory mapped, the write accesses by the command sequencer can be enabled by setting HSSPIn_CSCFG:SRAM = '1'.

The sequence of command phases (i.e. instruction phase, address phase and data phase) generated by the HSSPI in command sequencer mode on the SDATA lines is configured by the software (during initialization of the HSSPI) in the CSR.

Generation of memory read command sequence

For memory read the sequence of command phases can be configured in the list of eight registers: HSSPIn_RDCSDC0, HSSPIn_RDCSDC1, HSSPIn_RDCSDC2, HSSPIn_RDCSDC3, HSSPIn_RDCSDC4, HSSPIn_RDCSDC5, HSSPIn_RDCSDC6 and HSSPIn_RDCSDC7. Each of the eight registers is parsed in order, starting with HSSPIn_RDCSDC0 to HSSPIn_RDCSDC7. Refer to Figure 33-51.

Figure 33-51. Memory read command sequence list



The HSSPIn_RDCSDCO:DEC bit in each of these registers indicates whether the data byte (i.e. RDCSDATA[2:0]) must be decoded (as shown in Table 33-39) or whether the data byte in RDCSDATA[7:0] must be transmitted as it is.

.

Table 33-39. Decoding of the read command sequence list

HSSPIn_RDCSDCO :DEC	RDCSDATA [2:0]	Description
'0'	Don’t care	Transmit RDCSDATA[07:00] as it is
'1'	'000'	Transmit address bits [07:00] of the serial memory to be accessed
'1'	'001'	Transmit address bits [15:08] of the serial memory to be accessed
'1'	'010'	Transmit address bits [23:16] of the serial memory to be accessed
'1'	'011'	Transmit address bits [31:24] of the serial memory to be accessed
'1'	'100'	Tri-state the SDATA output lines, for one byte-time
'1'	'111'	End of list

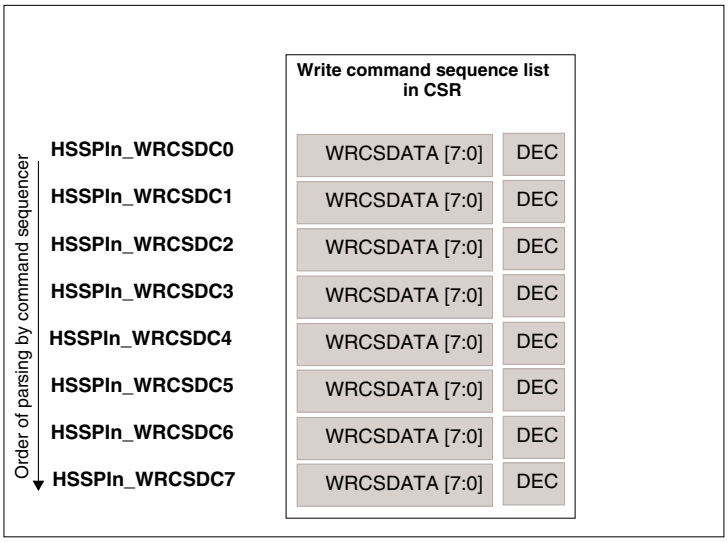
The command sequencer switches to data-read cycles if it gets 'end of list' or after the HSSPIn_RDCSDC7 register has been parsed whichever occurs first. During data read cycles the serial data on SDATA lines are sampled and the assembled data are returned to the AHB master in response to its AHB read transaction.

Generation of memory write command sequence

Memory write command sequences can be initiated by the command sequencer only if they are enabled by the HSSPIn_CSCFG:SRAM bit.

For memory write the sequence of command phases can be configured in the list of eight registers: HSSPIn_WRCSDC0, HSSPIn_WRCSDC1, HSSPIn_WRCSDC2, HSSPIn_WRCSDC3, HSSPIn_WRCSDC4, HSSPIn_WRCSDC5, HSSPIn_WRCSDC6, and HSSPIn_WRCSDC7. Each of the eight registers is parsed in order, starting with HSSPIn_WRCSDC0 to HSSPIn_WRCSDC7. Refer to Figure 33-52.

Figure 33-52. Memory write command sequence list



The HSSPIn_RDCSDCO:DEC bit in each of these registers indicates whether the data byte (i.e. WRCSDATA[2:0]) must be decoded (as shown in Table 33-40) or whether the data byte in WRCSDATA[7:0] must be transmitted as it is.

Table 33-40. Decoding of the write command sequence list

HSSPIn_ RDCSDCO :DEC	WRCSDATA [2:0]	Description
'0'	Don’t care	Transmit WRCSDATA[07:00] as it is
'1'	'000'	Transmit address bits [07:00] of the serial memory to be accessed
'1'	'001'	Transmit address bits [15:08] of the serial memory to be accessed
'1'	'010'	Transmit address bits [23:16] of the serial memory to be accessed
'1'	'011'	Transmit address bits [31:24] of the serial memory to be accessed
'1'	'100'	Tri-state the SDATA output lines, for one byte-time
'1'	'101'	Irrespective of the shift-direction configured in HSSPIn_PCC0~3:SDIR bit, the transmission takes place in the following order: 1. Transmit RDCSDATA[07:04] as it is. Transmit direction of this data depends on the value of HSSPIn_PCC0~3:SDIR bit. 2. Tri-state the SDATA output lines for 4-bit times.
'1'	'111'	End of list

The command sequencer switches to data write cycles if it gets 'end of list' or after the HSSPIn_WRCSDC7 register has been parsed whichever occurs first. During data write cycles, the parallel data from the AHB write data bus (i.e. HWDATA) is serially transmitted over the SDATA lines, as defined by the configured SPI protocol.

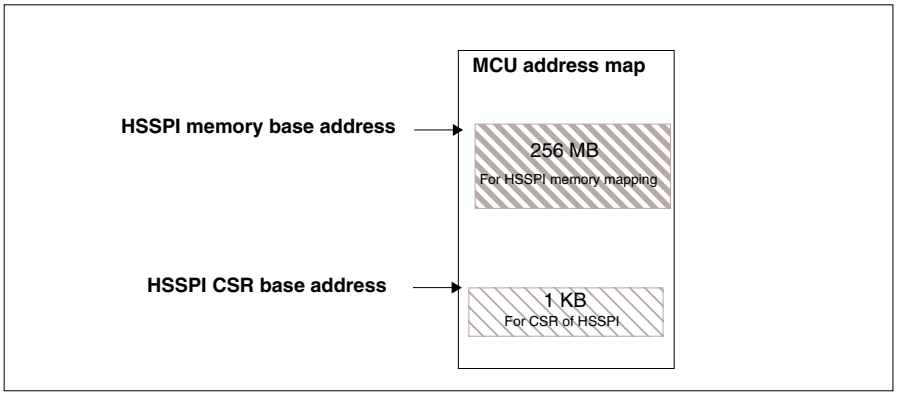
33.6 Address map of HSSPI

The HSSPI module is allocated 256 MB of the MCU's address space for memory mapping of the external serial memory devices in command sequencer mode. An additional 1 KB of the MCU's address space is reserved for mapping the internal Configuration and Status Registers (i.e. CSRs) of the HSSPI. The address area allocated to the HSSPI is shown in this section.

Arrangement of the HSSPI address space in memory

Figure 33-53 shows the allocation of the HSSPI address space in the MCU's address space.

Figure 33-53. Address map of the HSSPI



Allocation for memory mapped devices

The 256 MB of memory space, starting from 'HSSPI memory base address' is reserved for memory mapping of the external serial devices onto the MCU's address space.

This space is used in command sequencer mode.

Allocation for CSRs

The 1 KB of memory space, starting from 'HSSPI CSR base address' is reserved for memory mapping of the Configuration and Status Registers of the HSSPI onto the MCU's address space.

33.7 Notes on using HSSPI

This section is the 'programmer's guide', which lists the usage notes for programming the HSSPI module. Programmers should read these guidelines before programming the HSSPI module.

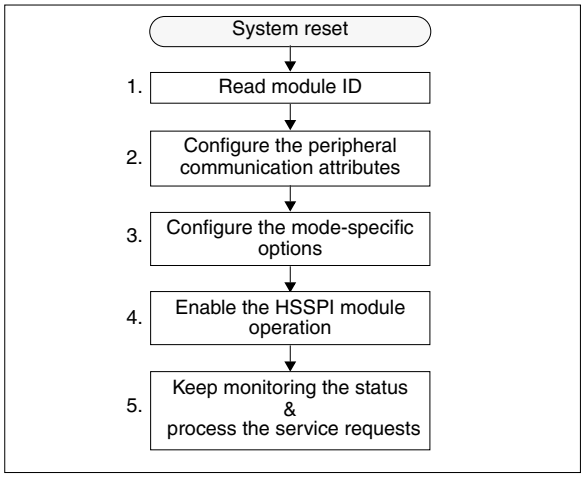
General usage notes

- Any serial-transaction-related parameters and control bits (e.g. selection of the attributes in the HSSPIn_PCC0~3 registers, switching between direct mode of operation and the command sequencer mode of operation, switching between the master and slave mode of operation, etc) should not be changed while a transaction is in progress. Any such changes are performed only after the HSSPI module is disabled (HSSPIn_MCTRL:MEN = '0') and the current serial transfer has ended (i.e. HSSPIn_TXF:TSSRS = '1' or HSSPIn_RXF:RSSRS = '1'). To ensure that the HSSPI module has finished all its transfers the software reads the HSSPIn_DMSTATUS:TXACTIVE and the HSSPIn_DMSTATUS:RXACTIVE bits
- To use the transfer protocols of certain serial flash memory devices (like the devices from Winbond™ in direct mode of operation, it is necessary to transfer an initial set of bytes in legacy mode and then transfer the remaining sets of data bytes using the dual-bit or quad-bit mode. In such cases, it might be necessary to change the HSSPIn_DMTRP register while a serial transfer is in progress (i.e. while one of the slave select lines: SSEL0~3 is asserted). When the software has to re-program the HSSPIn_DMTRP register while a serial transfer has already started, it can be done after halting the current serial transfer (as explained in 'Halting a transfer due to lack of TX DATA or due to lack of RX-FIFO space' in 33.4 Direct mode)
- However, it is recommended that while the serial transfer is halted, the HSSPIn_DMTRP register should not be reprogrammed for switching from 'TX only legacy' to 'RX only legacy' mode and vice-versa. Instead of halting the serial transfer for switching the transfer protocol from 'TX only legacy' to 'RX only legacy', the software can program the HSSPIn_DMTRP for 'TX and RX legacy' mode before the start of a transfer and then ignore the bytes in the data received while reception is not applicable, or transmit dummy data when transmission is not applicable
- When direct mode of operation is used, the software is responsible for ensuring that the internal FIFOs of HSSPI are not overrun or underrun. If an overrun or underrun occurs, the FIFO fill- levels (i.e. HSSPIn_DMSTATUS:TXFLEVEL and HSSPIn_DMSTATUS:RXFLEVEL) are no longer valid and the software must to flush the FIFOs Flushing the FIFOs beore using them for any serial transfer is done using the HSSPIn_FIFOCFG:RXFLSH and HSSPIn_FIFOCFG:TXFLSH bits.
- Flushing a FIFO ensures that they are not pre-loaded with any garbage data (possibly from the previous transfer)
- In direct mode and master operation whenever the transfer ends, the data from the RX shift register is pushed into the RX-FIFO provided that the RX-FIFO is not full: If the RX-FIFO is already full while a serial transfer is terminating, the serial transfer halts and any remaining data stays in the RX shift register as long as the HSSPI is halted. As soon as the RX-FIFO is no longer full, the, HSSPI comes out of the halt state and any remaining data from the RX shift register is pushed into the RX-FIFO before HSSPI releases the slave select line. Thus, in direct mode master operation, no data ever remains in the RX shift register. The bit- count from the HSSPIn_RXBITCNT register is always '0' and all received data is always pushed into the RX-FIFO
- When the transfer stops, in direct mode slave operation, the last data received by the HSSPI which has not yet and been pushed into the RX-FIFO remains in the RX shift register. The software should then read the remaining data from the HSSPIn_RXSHIFT register and the bit- count from the HSSPIn_RXBITCNT register

Steps in programming the HSSPI module

Figure 33-54 shows the general steps for using the HSSPI as a master in direct mode of operation.

Figure 33-54. Programmer's flowchart: general steps



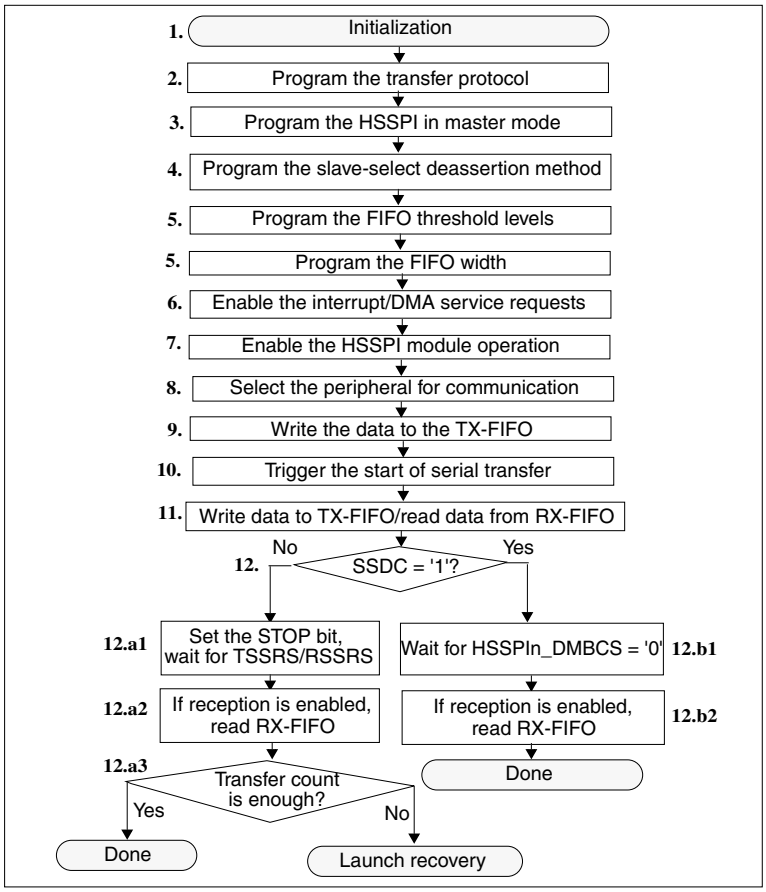
1. After system reset the software should detect the module ID number of the HSSPI by reading the HSSPIn_MID register. This would helps it to identify the attributes and capabilities supported by the HSSPI module.
2. The next step is to configure the attributes related to the peripheral communication with the serial device(s) connected with the HSSPI. In master mode the HSSPI can be interfaced with up to four serial devices. In slave mode the HSSPI can respond to the serial transactions initiated by a single master device. Serial communication-related attributes, such asclock polarity, clock phase, transfer frequency (i.e. clock division ratio and clock source selection bits), slave select polarity, etc. should be configured in the HSSPIn_PCC0, HSSPIn_PCC1, HSSPIn_PCC2, and HSSPIn_PCC3 registers. It is very important that these attributes are the same as those being used by the remote serial device with which the HSSPI is serially interfaced. These configurations should not be modified while the HSSPI module is active. In case the software has to re-program the values, it should first disable the HSSPI module and wait until the current serial transfer has finished.
3. The HSSPI can be configured either in direct mode or in command sequencer mode through the HSSPIn_MCTRL:CSEN bit. Depending on the mode is to be used, the software needs tol configure the mode-specific registers. The registers specific to the direct mode are: HSSPIn_TXF, HSSPIn_TXE, HSSPIn_TXC, HSSPIn_RXE, HSSPIn_RXF, HSSPIn_RXE, HSSPIn_RXC, HSSPIn_DMCFG, HSSPIn_DMSTATUS, HSSPIn_RXSHIFT, HSSPIn_TXFIFO0~15, HSSPIn_RXFIFO0~15, and HSSPIn_FIFOCFG. The registers specific to the command sequencer mode are: HSSPIn_CSCFG, HSSPIn_CSITIME, HSSPIn_CSAEXT, HSSPIn_RDCSDC0 - HSSPIn_RDCSDC7, and HSSPIn_WRCSDC0 - HSSPIn_WRCSDC7.
4. Only after all module-specific configurations are programmed, the HSSPI module shall be enabled (by setting the HSSPIn_MCTRL:MEN to '1').
5. Once enabled, the HSSPI module enters normal working mode. The software should constantly monitor the status of the HSSPI module using the various status bits. If the HSSPI module is configured to initiating the service requests (i.e interrupts and/or DMA requests), then these will be peiodically triggered. The software should service those requests in order to ensure the normal working of the HSSPI.

Using the HSSPI in direct mode of operation

Master mode

Figure 33-55 shows the general steps for using the HSSPI as a master in direct mode of operation.

Figure 33-55. Programmer's flowchart: HSSPI as a master, in direct mode of operation



1. After system reset the software should initialize the HSSPI module by reading the HSSPIn_MID register and setting the peripheral communication related attributes in the HSSPIn_PCC0, HSSPIn_PCC1n, HSSPIn_PCC2n and HSSPIn_PCC3 registers. It is very important that these attributes are the same as those used by the remote serial device with which the HSSPI is serially interfaced. Ensure that the HSSPIn_MCTRL:CSEN bit is reset to '0'.
2. Next configure the transfer protocol in the HSSPIn_DMTRP:TRP field. These bits determine if HSSPI serial transfers use the legacy, dual-bit or quad-bit SPI protocol, and if the HSSPI is used to transmit only, receive onlz or both.
3. Set the HSSPIn_DMCFG:MST bit to '1', this configures the HSSPI as a master device.
4. Then configure the its HSSPIn_DMCFG:SSDC field. This selects how the slave select output will be deasserted. If byte-counter mode is used, load the HSSPIn_DMBCC:BCC field with the number of bytes to be serially transferred. If software flow control is used, the software is responsible to set the HSSPIn_DMSTOP:STOP bit after it has finished transmiting/receiving of the desired data.
5. Configure the HSSPIn_FIFOCFG register to set the FIFO threshold levels. The programming of these levels conrol the assertion of the service requests. The width of FIFOs is determined by configuring the HSSPIn_FIFOCFG:FWIDTH field.
6. Configure the service requests. The HSSPI supports both interrupt and DMA service requests for the normal data read/write operations from/to the internal FIFOs. For normal operation, either the interrupt requests or the DMA requests are enabled by the software. To enable the interrupt service requests for TX-FIFO write requests, program the bits in the HSSPIn_TXE register. To enable the interrupt service requests for RX-FIFO read requests, program the bits in the HSSPIn_RXE register. To enable the DMA read and/or DMA write service requests, program either/both of the HSSPIn_DMDMAEN:TXDMAEN and the HSSPIn_DMDMAEN:RXDMAEN bits. The DMA read channel must be set up to perform a block transfer of 'HSSPIn_FIFOCFG:RXFTH + 1' transfers. The DMA write channel must be setup to perform a block transfer of '16 - HSSPIn_FIFOCFG:TXFTH' transfers.
7. The HSSPI is now initialized for direct mode of operation. Next set the HSSPIn_MCTRL:MEN bit to enable the module.

8. Select the peripheral (in the HSSPIn_DMPSEL:PSEL field) to which HSSPI shall initiate the transfer.
9. If the HSSPI is configured for TX only or TX and RX mode of operation (in the HSSPIn_DMTRP:TRP field), then write the data to be transmitted to the TX-FIFO by performing write accesses to the HSSPIn_TXFIFO0~15 register address. Before writing to the HSSPIn_TXFIFO0~15 register, modify the value of the HSSPIn_FIFOCFG:TXCTRL field appropriately. Generally (i.e when the data being written to the TX-FIFO is to be transmitted as it is) the HSSPIn_FIFOCFG:TXCTRL bit should be reset to '0'. Only when the HSSPI is to be instructed to tri-state the serial data lines for a byte-time or 4 bit-times should the HSSPIn_FIFOCFG:TXCTRL bit shall be set to '1' and a HSSPIn_TXFIFO0~15 write access performed.
10. When the HSSPI is configured as a master, setting the HSSPIn_DMSTART:START bit triggers the start of the serial transaction.
    If transmission is enabled in the HSSPIn_DMTRP:TRP field once the serial transaction starts, the HSSPI reads the TX-FIFO and loads the shift register. The shift register is shifted either left or right (based on configuration in HSSPIn_PCC0~3:SDIR field and the transmited data is shifted out onto the serial line(s). If the HSSPI is enabled to receive (in HSSPIn_DMTRP:TRP field), the HSSPI assembles the received data by serially shifting the received data into the shift register. The received data assembled in the shift register is shifted into the RX-FIFO.
11. Service requests are asserted by the HSSPI whenever the TX-FIFO level is below the threshold or the HSSPI RX-FIFO level is above the threshold. The software should first read/write the FIFOs to ensure the HSSPI is operationg normally. Once processed, the interrupt service requests are cleared by the software in the HSSPIn_TXC or the HSSPIn_RXC registers. DMA service requests are automatically cleared by the HSSPI.
12. To stop the serial transfers, the software can use one of the two modes (configured in HSSPIn_DMCFG:SSDC bit) - software flow control mode or the byte counter mode.

    In Software flow control mode:

    a. In software flow control mode the software waits until either of the HSSPIn_TXF:TSSRS or the HSSPIn_RXF:RSSRS flag is set, indicating that the slave select is released.
    b. If the HSSPIn_DMTRP register is configured to receive the software fetches the received data from the RX-FIFO.
    c. If the number of bytes of data transferred using the software flow control mode is insufficient the software launches its own recovery.

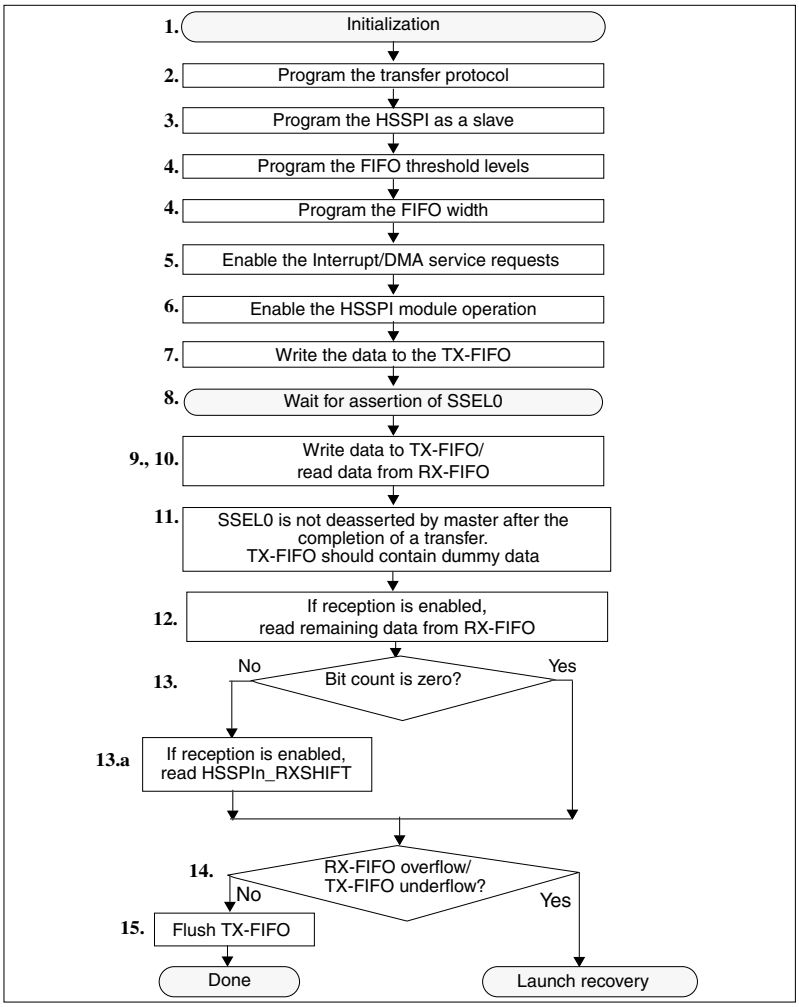
    In byte counter mode:

    a. In byte counter mode the software waits until the HSSPIn_DMBCS register value becomes '0'.
    b. Ifthe HSSPIn_DMTRP register is configured to receive the software fetches the received data from the RX-FIFO.
13. Normally the software repeats steps 9 to 12 or it loops back to the initialization step.
14. To switch from the direct mode of operation to the command sequencer mode or vice-versa, or change any of the parameters that directly affect the serial transfer, the software must first stop the current transfer and disable the HSSPI module. Only after it is ensured that the HSSPI module has finished all its transfers (by reading the HSSPIn_DMSTATUS:TXACTIVE and the HSSPIn_DMSTATUS:RXACTIVE bits) can the software can reconfigure the module.

Slave mode

Figure 33-56 shows the general steps for using the HSSPI as a slave in direct mode of operation.

Figure 33-56. Programmer's flowchart: HSSPI as a slave, in direct mode of operation



1. After system reset the software will initialize the HSSPI module by reading the HSSPIn_MID register and setting the peripheral communication related attributes in the HSSPIn_PCC0 register. It is very important that these attributes are the same as those used by the remote serial device with which the HSSPI is serially interfaced. Ensure that the HSSPIn_MCTRL:CSEN bit is set to '0'.
2. Next is to configure the transfer protocol in the HSSPIn_DMTRP:TRP field. These bits determine if HSSPI serial transfers use the legacy, dual-bit or quad-bit SPI protocol, and if the HSSPI is used to transmit only, receive only or both.
3. Then set the HSSPIn_DMCFG:MST bit to '0', This configures the HSSPI used as a slave device. If the HSSPI is used as a slave, it responds only to serial transfers initiated by the external master on its SSEL0 input. Therefore, when using the HSSPI in slave mode, only the peripheral communication attributes in HSSPIn_PCC0 register are used by the HSSPI.
4. Configure the HSSPIn_FIFOCFG register to set the FIFO threshold levels. The programming of these levels, control the assertion of the service requests. Also configure the HSSPIn_FIFOCFG:FWIDTH field to select the width of the FIFOs.
5. Configure the service requests. HSSPI supports both interrupt and DMA service requests for the normal data read/write operations from/to the internal FIFOs. For normal operation, either the interrupt requests or the DMA requests are enabled by the software. To enable the interrupt service requests for TX-FIFO write requests, program the bits in the HSSPIn_TXE register. To enable the interrupt service requests for RX-FIFO read requests, program the bits in the HSSPIn_RXE register. To enable the DMA read and/or DMA write service requests, program either/both of the HSSPIn_DMDMAEN:TXDMAEN and the HSSPIn_DMDMAEN:RXDMAEN bits. The DMA read channel must be set up to perform a block transfer of 'HSSPIn_FIFOCFG:RXFTH + 1' transfers. The DMA write channel must be setup to perform a block transfer of '16 - HSSPIn_FIFOCFG:TXFTH' transfers.

6. Set the HSSPIn_MCTRL:MEN bit to enable the module.
7. If the HSSPI is configured for TX only or TX and RX mode of operation (in the HSSPIn_DMTRP:TRP field), then write the data to be transmitted to the TX-FIFO by performing write accesses to the HSSPIn_TXFIFO0~15 register address. Before writing to the HSSPIn_TXFIFO0~15 register, modify the value of the HSSPIn_FIFOCFG:TXCTRL field appropriately. Generally (i.e when the data being written to the TX-FIFO is to be transmitted as it is), the HSSPIn_FIFOCFG:TXCTRL bit shall be reset to '0'. Only when the HSSPI is to be instructed to tri-state the serial data lines for a byte-time or for 4-bit times should the HSSPIn_FIFOCFG:TXCTRL bit shall be set to '1' and a HSSPIn_TXFIFO0~15 write access performed.
8. When the HSSPI is configured as a slave, it waits for the remote master to assert its SSEL0 input.
9. If transmission is enabled in the HSSPIn_DMTRP:TRP field once the serial transaction starts, the HSSPI reads the TX-FIFO and loads the shift register. The shift register is shifted either left or right (based on configuration in HSSPIn_PCC0~3:SDIR field and the transmitted data is shifted out onto the serial line(s). If the HSSPI is enabled to receive (in HSSPIn_DMTRP:TRP field), the HSSPI assembles the received data by serially shifting the received data into the shift register. The received data assembled in the shift register is shifted into the RX-FIFO.
10. Service requests are asserted by the HSSPI whenever the TX-FIFO level is below the threshold or the RX-FIFO level is above the threshold. The software then reads/writes the FIFOs, to ensure the HSSPI is operationg normally. Once processed, the interrupt service requests are cleared by the software in the HSSPIn_TXC or the HSSPIn_RXC registers. DMA service requests are automatically cleared by the HSSPI.
11. In slave operation, complete frames are transferred without SSEL de-selection

    Note: When the transfer ends, the HSSPIn_TXFIFO registers should not be empty. The remaining data is dummy data (at least one dummy data is necessary).
12. If enabled to receive, the software will read the received data from the RX-FIFO.
13. When the transfer stops, in slave mode of operation, the remaining data received by the HSSPI, which has not yet been pushed into the RX-FIFO remains in the RX shift register and the number of bits valid in the RX shift register is indicated by the RX bit count register HSSPIn_RXBITCNT.If the value in HSSPIn_RXBITCNT is non-zero, the software should read the remaining data from the HSSPIn_RXSHIFT register.
14. In slave mode, the software must ensure that the RX-FIFO does not overflow and that the TX- FIFO does not underrun. If either of these occurs, the software must launch its recovery operation.
15. Once a transfer has completed, the HSSPIn_TXFIFO registers must be flushed.
16. Normally the software usually keeps repeats steps 8 to 15, or it loops back to the initialization step.
17. To switch between the direct mode of operation and the command sequencer mode (note, that in this case, switching to command sequencer mode also involves switching from slave mode of operation to master-mode of operation) or to reprogram any of the parameters that directly affect the serial transfer, the software must first stop the current transfer, and disable the HSSPI module. Only after it is ensured that the HSSPI module has finished all its transfers (by reading the HSSPIn_DMSTATUS:TXACTIVE and the HSSPIn_DMSTATUS:RXACTIVE bits) should the software reconfigure the module.

Using the memory mapped memories

Following rules should be followed when interfacing serial memories, for memory-mapped accesses in command sequencer mode.

Usage rules and notes

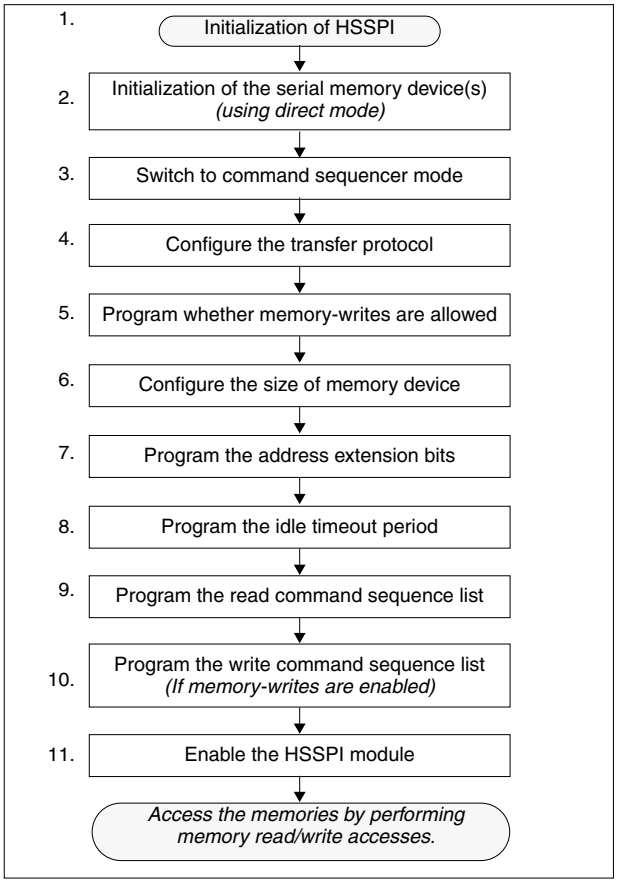
- In command sequencer mode, all serial memory devices interfaced with the HSSPI should be from same family. Do not mix serial memory devices from different vendor families
- If memory devices from same family but with different memory sizes are to be interfaced, then while determining the suitable value of the HSSPIn_CSCFG:MSEL field, the memory device with maximum size must be considered. Note: That the number of bytes from the final memory address that will be transmitted by the HSSPI to the serial memory-mapped device is programmed in the command sequence lists (in HSSPIn_RDCSDC0~7 and HSSPIn_WRCSDC0~7 registers). Interfacing a memory device which requires 32-bit addressing with one from the same family but rtequires 24-bit addressing in command sequencer mode is not possible. This is because a memory device with 24-bit addressing cannot be used with a bit- stuffed 32-bit address, its address phase is of 3 cycles only
- If a memory device only needs 21-bit addressing, and if the command sequence in the HSSPI is configured to transmit a 24-bit address to the memory device, then the three most significant bits [23:21] are bit-stuffed with '0's (using the HSSPIn_CSAEXT register) by the software. If the unused most significant bits are not reset to '0's, then the address pointers in the serial memory devices interfaced with the HSSPI might wrap, causing unwanted results

- Serial SRAM devices support burst operation only when they are configured to work in burst mode. However, the command sequencer always assumes the SRAM device is in burst mode. therefore before enabling the command sequencer mode of the HSSPI, the software needs to configure the serial SRAM device (using direct mode of operation) for burst mode of operation
- In command sequencer mode it is not possible to change between the legacy, dual-bit or quad- bit modes when a transfer has started. This means that for some of the new serial flash devices, such as those from Winbond™, the command sequencer can be enabled only after the memory device has been initialized to work in the 'continuous read mode', which can be done using the direct mode of operation of HSSPI.

Programmer's flowchart

Figure 33-57 shows the general steps needed to use the HSSPI to memory mapping the serial memory devices onto the address space of the MCU.

Figure 33-57. Programmer's flowchart: memory mapping of serial memory devices



1. After system reset the software will initialize the HSSPI module by setting the peripheral communication related attributes in the Peripheral Communication Configuration Registers HSSPIn_PCC0, HSSPIn_PCC1n, HSSPIn_PCC2n, and HSSPIn_PCC3 registers. It is very important that these attributes are the same as being used by the remote serial device with which the HSSPI is serially interfaced. When serial memory devices are to be memory mapped using command sequencer mode, all memory devices should be from the same family. Therefore, all four Peripheral Communication Configuration Registers (i.e HSSPIn_PCC0~3) willl have same configuration values.
2. The next step is to initialize the serial device that is to be memory mapped. The initialization is device specific and it may include setting some control/status bits in its register set. e.g. to use a quad-SPI serial memory device, the device may need to be set to high-performance (i.e. quad mode). Consult the datasheet of the serial memory device to be interfaced. This initialization of the serial memory device is performed in direct mode of HSSPI.

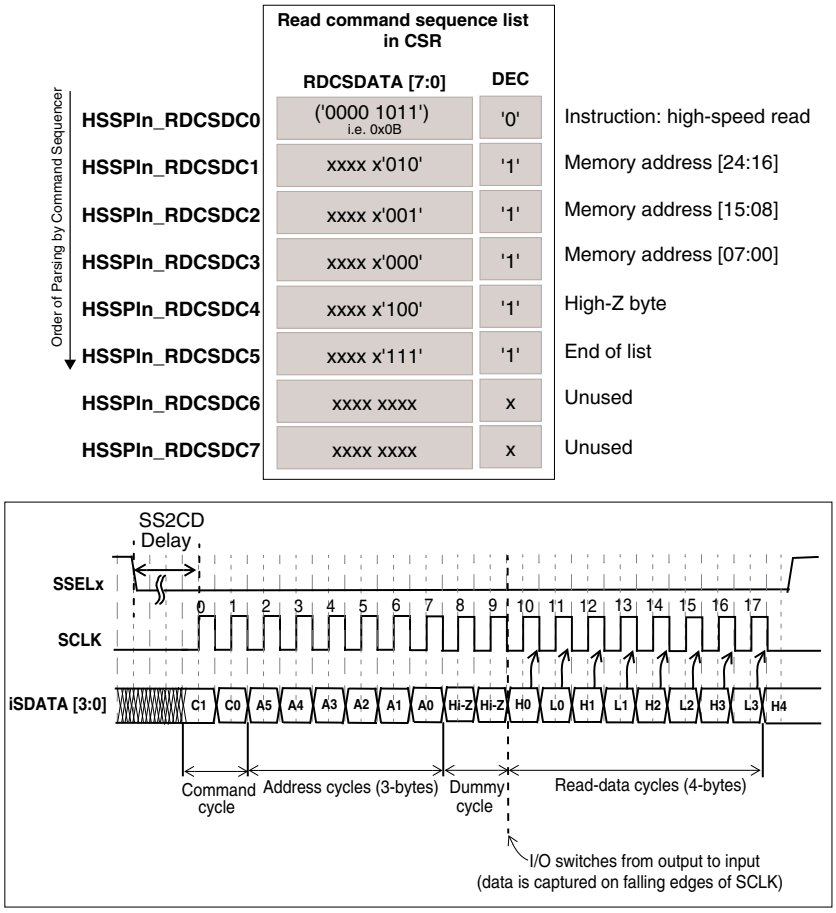
3. After initializing the serial device (in the direct mode of operation), re-program the HSSPI module in the command sequencer mode. To switch from one mode to the other, the software must first stop the current transfer and disable the HSSPI module. Only after it is ensured that the HSSPI module has finished all its transfers (by reading the HSSPIn_DMSTATUS:TXACTIVE and the HSSPIn_DMSTATUS:RXACTIVE bits), the software can reconfigure the module to command sequencer mode.
4. The next step is to configure the transfer protocol (i.e. whether the HSSPI serial transfers use the legacy, dual-bit or the quad-bit SPI protocol) in the HSSPIn_CSCFG:MBM field.
5. If serial SRAM devices are connected, enable the write accesses to these memory-mapped devices using the HSSPIn_CSCFG:SRAM bit. If serial flash devices are connected, do not enable the write accesses.
6. Program the HSSPIn_CSCFG:MSEL field with the size of the AHB address space which must be used to select of the memory device on which the serial transfer must be initiated. For details of the slave selection logic refer to 33.5 Command sequencer mode.
7. If the addresses generated for the memory-mapped accesses are to be virtually extended to cover a memory range of virtually 16 GB, program the HSSPIn_CSAEXT register. For details of address generation refer to 33.5 Command sequencer mode.
8. The HSSPIn_CSITIME:ITIME field is used to enhance the overall performance of the memory- mapped accesses by continuing the previous serial transaction for a configurable period. If there is an access (of the same type: i.e. read/write) to the consecutive memory address, HSSPI proceeds with the same serial transfer (without having to issue a new command/address cycles), thereby reducing the access time.
9. Program the HSSPIn_CSITIME:ITIME with an appropriate idle time-out value. Program the list of Read Command Sequence Registers (i.e. HSSPIn_RDCSDC0, HSSPIn_RDCSDC1, HSSPIn_RDCSDC2, HSSPIn_RDCSDC3, HSSPIn_RDCSDC4, HSSPIn_RDCSDC5, HSSPIn_RDCSDC6, and HSSPIn_RDCSDC7) with the sequence of the memory read commands for the memory device being interfaced. Consult with the device specific datasheet for details of the read command sequence.
10. If memory-write accesses are also enabled in the HSSPIn_CSCFG:SRAM bit, the list of Write Command Sequence Registers have to be programmed (i.e. HSSPIn_WRCSDC0, HSSPIn_WRCSDC1, HSSPIn_WRCSDC2, HSSPIn_WRDCSDC3n, HSSPIn_WRCSDC4, HSSPIn_WRCSDC5, HSSPIn_WRCSDC6, and HSSPIn_WRCSDC7) with the sequence of the memory write commands for the memory device that is being interfaced. Consult with the device specific datasheet for details of the write command sequence.
11. Enable the HSSPI module (in the HSSPIn_MCTRL:MEN bit) so that it starts generating the read/write sequences on the serial interface by mapping the AHB accesses to the memory- mapped locations.

Timing diagram for command sequencer

Figure 33-58 illustrates with an example, how the command sequencer generates the serial memory read command sequence. Assume that the read command sequence list is programmed in HSSPIn_RDCSDC0 through the HSSPIn_RDCS5 registers as shown in the figure. The command sequencer parses the list, starting from the HSSPIn_RDCS0 register, and executes the commands as described in 33.5 Command sequencer mode.

Figure 33-58 shows the corresponding timing diagram for a read command sequence for a clocking mode of '0'.

Figure 33-58. Read command sequence illustration with timing diagram (mode-0)



34. Inter IC Sound (I^2S)

This chapter explains the functions and operations of the serial audio interface that is the Inter IC Sound (I2S).

34.1 Outline of the I^2S

This section describes the features and the block diagram of the I2S module.

I2S module is a full duplex, synchronous serial audio interface for multi channel specification. It can be configured to various frame formats by register setting.

This module can be set to operate as master and slave. In the master mode, clock (SCK) and frame synchronous signal (WS) are output to the external slave. In the slave mode, they are input from the external master.

During the master mode, SCK clock can be output by dividing external clock or internal clock (it is selectable by register). Frame synchronous signal can be generated by free-running or burst mode (generated only when there is transmission data).

This module has transmission/reception FIFOs, and their depths depend on the mode:

In transmission only mode, there is a 132-word 32-bit transmission FIFO. In reception only mode, there is a 132-word 32-bit reception FIFO. This module can also be configured in simultaneous mode. Simultaneous mode operates with a 66-word 32-bit transmission FIFO and a 66-word 32-bit reception FIFO.

Internal transfer between transmission and reception FIFO and internal system memory can be performed by DMA, interrupt, and polling.

Features of I2S

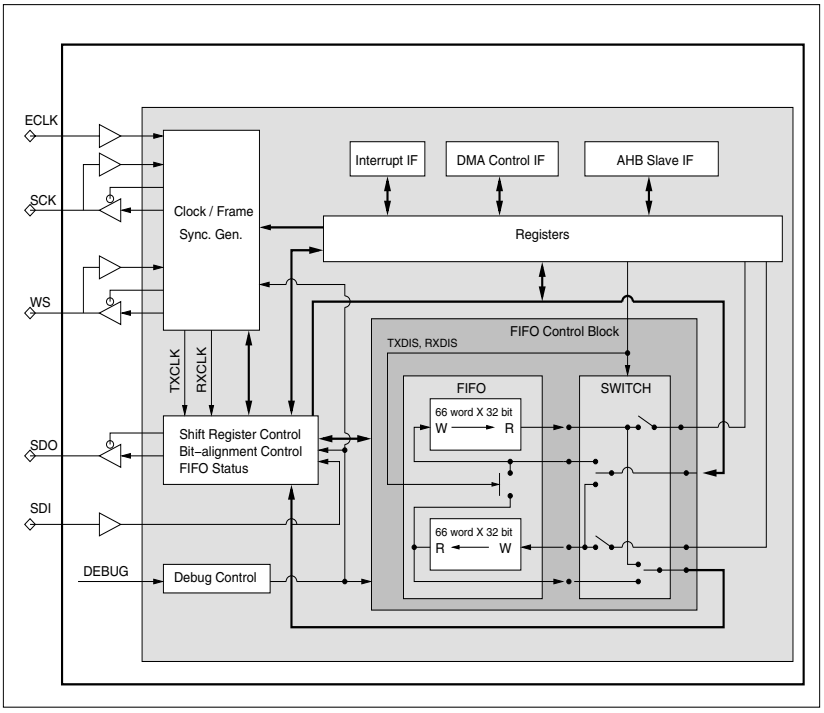
I2S interface has the following features:

- Programmable master/slave operations
- Support of transmission only, reception only and simultaneous transmission/reception modes
- Selecting 1 sub frame and 2 sub frame constructions
- Setting up to 32 channels to each sub frame
- Individually setting number of channel in each sub frame
- Individually setting channel length of each sub frame (number channel bit)
- Individually setting word length in channel of each sub frame
- Setting valid/invalid of each channel in each sub frame.
- Data is not sent or received to invalid channel
- Setting word length from 7 to 32 bits
- Programming frequency of frame synchronous signal
- Setting up to 3071 bits in 1 frame
- Programming width of frame synchronous signal (1 bit or 1 channel length)
- Programming phase of frame synchronous signal (0-bit or 1-bit delay)
- Setting polarity of frame synchronous signal
- Setting polarity of serial bit clock
- Programming sampling point of received data (center or at the end of received data)

- Selecting clock frequency source of serial bit clock in the master mode (internal and external clock)
- Setting clock frequency dividing ratio in the master mode
- Frequency of SCK = (frequency of internal clock or external clock) / (2 x I2Sn_CNTREG:CKRT[5:0])
- Frequency dividing ratio is settable within 0-126 in multiple of 2 (when the ratio is '0', frequency dividing source is bypassed)
- Data transfer to system memory by DMA, interrupt, and polling
- PPU support
- Debug support

Block diagram of I2S

Figure 34-1. Block diagram of I2S



- Four wire interface is used for full duplex data transfer (separate line for serial data input and serial data output)
- Frame and clock lines are bi-directional. These lines are output when module is configured in master mode and act as input line in slave mode
- DMA controller is used for DMA access

Clocking of I2S

- The supply clock of I2S can be internal (CLK_PERI4_PD2) or external (ECLK) source. For details refer to device specific datasheet. This clock is then prescaled to required frequency through I2S control register bits I2Sn_CNTREG:CKRT[5:0]
- Frame frequency can be adjusted using I2Sn_CNTREG:OVHD[9:0] bits

34.2 I2S registers

This section describes the registers of the I2S in detail.

The suffix 'n' in the register name indicates that the register is in instance 'n' of the module.

Registers of I2S

The following registers are available for each instance of I2S:

- Reception FIFO Data Register (I2Sn_RXFDAT0~15)
- Transmission FIFO Data Register (I2Sn_TXFDAT0~15)
- Control Register (I2Sn_CNTREG)
- Channel Control Register 0 (I2Sn_MCR0REG)
- Channel Control Register 1 (I2Sn_MCR1REG)
- Channel Control Register 2 (I2Sn_MCR2REG)
- Operation Control Register (I2Sn_OPRREG)
- Software Reset Register (I2Sn_SRST)
- Interrupt Control Register (I2Sn_INTCNT)
- Status Register (I2Sn_STATUS)
- DMA Activate Register (I2Sn_DMAACT)
- Debug Register (I2Sn_DEBUG)
- Module ID Register (I2Sn_MIDREG)

Memory layout of I2S registers

Table 34-1. Memory layout of I2S registers

Offset	+3	+2	+1	+0
0x00000000	I2Sn_RXFDAT0 00000000 00000000 00000000 00000000			
0x00000004	I2Sn_RXFDAT1 00000000 00000000 00000000 00000000			
0x00000008	I2Sn_RXFDAT2 00000000 00000000 00000000 00000000			
0x0000000C	I2Sn_RXFDAT3 00000000 00000000 00000000 00000000			
0x00000010	I2Sn_RXFDAT4 00000000 00000000 00000000 00000000			
0x00000014	I2Sn_RXFDAT5 00000000 00000000 00000000 00000000			
0x00000018	I2Sn_RXFDAT6 00000000 00000000 00000000 00000000			
0x0000001C	I2Sn_RXFDAT7 00000000 00000000 00000000 00000000			
0x00000020	I2Sn_RXFDAT8 00000000 00000000 00000000 00000000			

Table 34-1. Memory layout of I2S registers

Offset	+3	+2	+1	+0
0x00000024	I2Sn_RXFDAT9 00000000 00000000 00000000 00000000			
0x00000028	I2Sn_RXFDAT10 00000000 00000000 00000000 00000000			
0x0000002C	I2Sn_RXFDAT11 00000000 00000000 00000000 00000000			
0x00000030	I2Sn_RXFDAT12 00000000 00000000 00000000 00000000			
0x00000034	I2Sn_RXFDAT13 00000000 00000000 00000000 00000000			
0x00000038	I2Sn_RXFDAT14 00000000 00000000 00000000 00000000			
0x0000003C	I2Sn_RXFDAT15 00000000 00000000 00000000 00000000			
0x00000040	I2Sn_TXFDAT0 00000000 00000000 00000000 00000000			
0x00000044	I2Sn_TXFDAT1 00000000 00000000 00000000 00000000			
0x00000048	I2Sn_TXFDAT2 00000000 00000000 00000000 00000000			
0x0000004C	I2Sn_TXFDAT3 00000000 00000000 00000000 00000000			
0x00000050	I2Sn_TXFDAT4 00000000 00000000 00000000 00000000			
0x00000054	I2Sn_TXFDAT5 00000000 00000000 00000000 00000000			
0x00000058	I2Sn_TXFDAT6 00000000 00000000 00000000 00000000			
0x0000005C	I2Sn_TXFDAT7 00000000 00000000 00000000 00000000			
0x00000060	I2Sn_TXFDAT8 00000000 00000000 00000000 00000000			
0x00000064	I2Sn_TXFDAT9 00000000 00000000 00000000 00000000			
0x00000068	I2Sn_TXFDAT10 00000000 00000000 00000000 00000000			
0x0000006C	I2Sn_TXFDAT11 00000000 00000000 00000000 00000000			

Table 34-1. Memory layout of I2S registers

Offset	+3	+2	+1	+0
0x00000070	I2Sn_TXFDAT12 00000000 00000000 00000000 00000000			
0x00000074	I2Sn_TXFDAT13 00000000 00000000 00000000 00000000			
0x00000078	I2Sn_TXFDAT14 00000000 00000000 00000000 00000000			
0x0000007C	I2Sn_TXFDAT15 00000000 00000000 00000000 00000000			
0x00000080	I2Sn_CNTREG 00000000 00000000 00000000 01100000			
0x00000084	I2Sn_MCR0REG 00000000 00000000 00000000 00000000			
0x00000088	I2Sn_MCR1REG 00000000 00000000 00000000 00000000			
0x0000008C	I2Sn_MCR2REG 00000000 00000000 00000000 00000000			
0x00000090	I2Sn_OPRREG 00000000 00000000 00000000 00000000			
0x00000094	I2Sn_SRST 00000000 00000000 00000000 00000000			
0x00000098	I2Sn_INTCNT 01111111 00111111 00000000 00000000			
0x0000009C	I2Sn_STATUS 00000000 00000000 00000000 00000000			
0x000000A0	I2Sn_DMAACT 00000000 00000000 00000000 00000000			
0x000000A4	I2Sn_DEBUG 00000000 00000000 00000000 00000000			
0x000000A8	I2Sn_MIDREG 00000000 00000000 00000000 00000000			
0x000000AC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			

34.2.1 Reception FIFO Data Register (I2Sn_RXFDAT0~15)

These registers are reception FIFO registers that can maintain up to 66 words (simultaneous transfer mode) or 132 words (reception only mode). There are 16 such registers, all consecutively placed in the register map. This is to support AHB burst transfers. Read access to any of the I2Sn_RXFDAT0~15 registers returns the word from reception FIFO. Each of these 16 registers are identical in function. Only one register (i.e. I2Sn_RXFDAT0) is explained here.

Reception FIFO Data Register 0 (I2Sn_RXFDAT0)

Figure 34-2. Reception FIFO Data Register 0 (I2Sn_RXFDAT0)

I2Sn_RXFDAT0																															
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**
RXDATA[31]	RXDATA[30]	RXDATA[29]	RXDATA[28]	RXDATA[27]	RXDATA[26]	RXDATA[25]	RXDATA[24]	RXDATA[23]	RXDATA[22]	RXDATA[21]	RXDATA[20]	RXDATA[19]	RXDATA[18]	RXDATA[17]	RXDATA[16]	RXDATA[15]	RXDATA[14]	RXDATA[13]	RXDATA[12]	RXDATA[11]	RXDATA[10]	RXDATA[9]	RXDATA[8]	RXDATA[7]	RXDATA[6]	RXDATA[5]	RXDATA[4]	RXDATA[3]	RXDATA[2]	RXDATA[1]	RXDATA[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-2. Reception FIFO Data Register 0 (I2Sn_RXFDAT0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	RXDATA	Receive Data The word received from serial bus is written to reception FIFO. When frame is 1 sub frame construction and word length set to I2Sn_MCR0REG:S0WDL is 32 bits or less (16 bits when I2Sn_CNTREG:RHLL register is '1'), it is written to reception FIFO after higher order bit is extended. When frame is 2 sub frame construction and word length set to I2Sn_MCR0REG:S0WDL is 32 bits or less (16 bits when I2Sn_CNTREG:RHLL register is '1'), reception data of sub frame '0' is written to reception FIFO after higher order bit is extended. For the case that word length set to I2Sn_MCR0REG:S1WDL is 32 bits or less, reception data of sub frame 1 is written to reception FIFO after higher order bit is extended. When I2Sn_CNTREG:BEXT is '1', it is extended with MSB of reception word (sign extension). For the case that the value is '0', it is enhanced by '0'. Reading this register returns a word of data from the Rx FIFO location pointed by the Rx FIFO read pointer. After a read access to this register, the Rx FIFO read pointer is incremented, provided that the read cycle was initiated by the AHB master other than the Debug Access Port (DAP) controller. If the DAP controller reads this register, the Rx FIFO read pointer is not incremented. When I2Sn_STATUS:RXNUM is 0x00000000, invalid data is read. An Rx FIFO underflow error (I2Sn_STATUS:RXUDR) flag is set, if the read cycle was initiated by the AHB master other than the DAP controller. If the DAP controller reads this register while the Rx FIFO is empty, the I2Sn_STATUS:RXUDR flag is not set. Non DAP access in non-privileged mode can only be done with full access rights i.e. iPPU_ACCESS = '1', else causes PPU error. For DAP access in non-privileged mode can be done with iPPU_ACCESS = '0' and iPPU_READ = '1'. Writing to I2Sn_RXFDAT0 generates PPU error.

34.2.2 Transmission FIFO Data Register (I2Sn_TXFDAT0~15)

These registers are transmission FIFO registers that can maintain up to 66 words (simultaneous transfer mode) or 132 words (transmission mode only). There are 16 such registers, all consecutively placed in the register map. This is to support AHB burst transfers. A write access to any of I2Sn_TXFDAT0~15 register writes a word of data in the transmission FIFO. Each of these 16 registers are identical in function. Only one register (i.e. I2Sn_TXFDAT0) is explained here.

Transmission FIFO Data Register 0 (I2Sn_TXFDAT0)

Figure 34-3. Transmission FIFO Data Register 0 (I2Sn_TXFDAT0)

I2Sn_TXFDAT0																															
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**
TXDATA[31]	TXDATA[30]	TXDATA[29]	TXDATA[28]	TXDATA[27]	TXDATA[26]	TXDATA[25]	TXDATA[24]	TXDATA[23]	TXDATA[22]	TXDATA[21]	TXDATA[20]	TXDATA[19]	TXDATA[18]	TXDATA[17]	TXDATA[16]	TXDATA[15]	TXDATA[14]	TXDATA[13]	TXDATA[12]	TXDATA[11]	TXDATA[10]	TXDATA[9]	TXDATA[8]	TXDATA[7]	TXDATA[6]	TXDATA[5]	TXDATA[4]	TXDATA[3]	TXDATA[2]	TXDATA[1]	TXDATA[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-3. Transmission FIFO Data Register 0 (I2Sn_TXFDAT0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TXDATA	Transmit Data Word to be transmitted can be written as long as transmission FIFO is not full. Write access can be performed regardless of shift register's operation status. Write access to full transmission FIFO is ignored and I2Sn_STATUS:TXOVR flag is set. Although writing data is accessed in word, half-word, and byte access, actual number of bits to be transmitted is determined by I2Sn_MCR0REG:S0WDL and I2Sn_MCR0REG:S1WDL (when frame is 2 sub frame). The data read from I2Sn_TXFDAT0 returns '0'.

34.2.3 Control Register (I2Sn_CNTREG)

The Control Register (I2Sn_CNTREG) sets the module configuration.

Control Register (I2Sn_CNTREG)

Figure 34-4. Control Register (I2Sn_CNTREG)

I2Sn_CNTREG																															
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**
CKRT[5]	CKRT[4]	CKRT[3]	CKRT[2]	CKRT[1]	CKRT[0]	OVHD[9]	OVHD[8]	OVHD[7]	OVHD[6]	OVHD[5]	OVHD[4]	OVHD[3]	OVHD[2]	OVHD[1]	OVHD[0]	read0	MSKB	MSMD	SBFN	RHLL	ECKM	BEXT	FRUN	MSLB	TXDIS	RXDIS	SMPL	CPOL	FSPH	FSLN	FSPL
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 34-4. Control Register (I2Sn_CNTREG) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	CKRT	Clock Divider This sets output clock frequency dividing ratio at master operation. Internal clock (CLK_PERI4_PD2) is divided at I2Sn_CNTREG:ECKM = '0', and external clock (ECLK) is divided at I2Sn_CNTREG:ECKM = '1'. Only even number of the ratio is supported and output clock's duty becomes 50%. '000000': Output clock is not divided '000001': Output clock is divided by 2 '000010': Output clock is divided by 4 '000011': Output clock is divided by 6 ... ... '111110': Output clock is divided by 124 '111111': Output clock is divided by 126

Table 34-4. Control Register (I2Sn_CNTREG) bits

Bit Position	Bit Field Name	Bit Description
[25:16]	OVHD	Frame Rate Control Frame rate can be adjusted by inserting overhead (OVHD) bits following the valid data of the frame. The overhead section of the transmission frame enters the state of high impedance. Up to 1023 overhead bits can be inserted at the end of the frame. The value set to OVHD becomes the number of insertion bits. The following expressions are formed for OVHD and frame synchronous signal cycle. 1 sub frame construction: • OVHD = Frame synchronous signal cycle/SCK cycle - (I2Sn_MCR0REG:S0CHL + 1) * (I2Sn_MCR0REG:S0CHN + 1) 2 sub frame construction: • OVHD = Frame synchronous signal cycle/SCK cycle - (I2Sn_MCR0REG:S0CHL + 1) * (I2Sn_MCR0REG:S0CHN + 1) - (I2Sn_MCR0REG:S1CHL + 1) * (I2Sn_MCR0REG:S1CHN + 1)
[15]	read0	-
[14]	MSKB	Serial Output Data in case of Invalid/Empty Frame Transmission For master operation (I2Sn_CNTREG:MSMD = '1'), free-running mode (I2Sn_CNTREG:FRUN = '0'), and I2Sn_OPRREG:TXENB = '1': ■ When transmission FIFO is empty at frame synchronous signal output, MSKB is output to all valid channels of its transmission frame. For slave operation (I2Sn_CNTREG:MSMD = '0') and I2Sn_OPRREG:TXENB = '1': ■ When transmission FIFO is empty at frame synchronous signal reception, MSKB is output to all valid channels of its transmission frame. For the case that transmission word length is shorter than the channel length, MSKB is driven to the rest of bit in transmission channel (channel length - word length).
[13]	MSMD	Master and Slave Mode Select Master and Slave modes are set. '0': Slave operation '1': Master operation

Table 34-4. Control Register (I2Sn_CNTREG) bits

Bit Position	Bit Field Name	Bit Description
[12]	SBFN	Sub Frame Construction Sub frame construction (number of sub frame) of the frame is specified. '0': 1 Sub frame construction (only sub frame 0) '1': 2 Sub frame construction (sub frame 0 and sub frame 1). Frame starts from 0th sub frame
[11]	RHLL	Word Construction Word construction of FIFO – 1 word (32 bits) or 2 half words (16 bits) – is specified. It is considered to be used at protocol, such as MSB-justified. '0': 32-bit FIFO word is handled as 1 word '1': 32-bit FIFO word is handled as 2 half words at serial bus with dividing 16 bits each to low order and high order. They are transferred by serial bus in order of low order, high order, low order, and high order. At reception, 2 consecutive half words from serial bus are handled as low order and high order, and they are put in 1 word (32 bits) to write to reception FIFO
[10]	ECKM	Clock Selector Clock frequency dividing is selected in master mode. '0': Internal clock (CLK_PERI4_PD2)is divided and output '1': External clock (ECLK)is divided and output
[9]	BEXT	Bit Extension When reception word length is shorter than the word length of FIFO (32 bits when RHLL is '0' and 16 bits when RHLL is '1'), extension mode of upper bit (word length of FIFO-reception word length) should be set. '0': Extended by '0' '1': Extended by sign bit (if MSB of word/half word is '1', then it is extended by '1', if MSB is '0' then it is extended by '0')
[8]	FRUN	Output Mode of Frame Synchronous Signal '0': Burst mode When I2Sn_OPRREG:START bit is '1', frame synchronous signal is output according to I2Sn_OPRREG:TXENB, I2Sn_OPRREG:RXENB, and transmission/reception FIFO conditions. When I2Sn_OPRREG:START bit is '0', frame synchronous signal is not output. '1': Free-running mode When I2Sn_OPRREG:START register is '1', frame synchronous signal proceeds free-running with the set frame rate. When I2Sn_OPRREG:START bit is '0', frame synchronous signal is not output.

Table 34-4. Control Register (I2Sn_CNTREG) bits

Bit Position	Bit Field Name	Bit Description
[7]	MSLB	Shifting Order Word bit's shift order is set. '0': Shift starts from MSB of the word '1': Shift starts from LSB of the word
[6]	TXDIS	Transmitter Disable Transmitting function is enabled or disabled. '0': Transmitting function is enabled '1': Transmitting function is disabled
[5]	RXDIS	Receiver Disable Receiving function is enabled or disabled. '0': Receiving function is enabled '1': Receiving function is disabled
[4]	SMPL	Sampling Point Sampling point of the data is specified. '0': Sampling at the center of reception data '1': Sampling at the end of reception data
[3]	CPOL	Clock Polarity SCK polarity which drives/samples serial data is specified. '0': Data is driven at rising edge of SCK, and sampled at falling edge '1': Data is driven at falling edge of SCK, and sampled at rising edge
[2]	FSPH	Frame Sync Phase Phase is specified to WS frame data. '0': WS becomes valid '1' clock before the first bit of frame data '1': WS becomes valid at the same time as the first bit of frame data
[1]	FSLN	Frame Sync Pulse Width Pulse width of WS is specified. '0': Pulse width is 1 cycle/SCK long (1-bit) '1': Pulse width is 1 channel long (1 channel) Pulse width of one channel FSLN = '1' is prohibited when frame length is set to one channel by I2Sn_MCR0REG:S0CHN = 0x0 and I2Sn_CNTREG:SBFN = '0'.
[0]	FSPL	Frame Sync Polarity Polarity of WS is set. '0': Frame synchronous signal becomes valid when WS is '1' '1': Frame synchronous signal becomes valid when WS is '0'

34.2.4 Channel Control Register 0 (I2Sn_MCR0REG)

The Channel Control Register 0 (I2Sn_MCR0REG) controls the frame construction.

Channel Control Register 0 (I2Sn_MCR0REG)

Figure 34-5. Channel Control Register 0 (I2Sn_MCR0REG)

I2Sn_MCR0REG																															
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**
read0	S1CHN[4]	S1CHN[3]	S1CHN[2]	S1CHN[1]	S1CHN[0]	S1CHL[4]	S1CHL[3]	S1CHL[2]	S1CHL[1]	S1CHL[0]	S1WDL[4]	S1WDL[3]	S1WDL[2]	S1WDL[1]	S1WDL[0]	read0	S0CHN[4]	S0CHN[3]	S0CHN[2]	S0CHN[1]	S0CHN[0]	S0CHL[4]	S0CHL[3]	S0CHL[2]	S0CHL[1]	S0CHL[0]	S0WDL[4]	S0WDL[3]	S0WDL[2]	S0WDL[1]	S0WDL[0]
Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-5. Channel Control Register 0 (I2Sn_MCR0REG) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30:26]	S1CHN	Sub Frame 1 Channel Numbers Number of channels of sub frame 1 is set. This is valid only when the frame is 2 sub frame construction (I2Sn_CNTREG:SBFN is '1') and is invalid when the frame is 1 sub frame construction (I2Sn_CNTREG:SBFN is '0'). Up to 32 channels can be specified, and S1CHN needs to be set to 'number of channel - 1'. Setting examples are shown below. '00000': Sub frame 1 becomes 1 channel construction '00001': Sub frame 1 becomes 2 channel construction '00010': Sub frame 1 becomes 3 channel construction ... ... '11110': Sub frame 1 becomes 31 channel construction '11111': Sub frame 1 becomes 32 channel construction

Table 34-5. Channel Control Register 0 (I2Sn_MCR0REG) bits

Bit Position	Bit Field Name	Bit Description
[25:21]	S1CHL	Sub Frame 1 Channel Length Channel length of the channel constructing sub frame 1 (bit length of channel) is set. 7 to 32 bits of channel length are available but 1 to 6 bits are prohibited. S1CHL needs to be set to 'channel length - 1' Setting examples are shown below. '00000'-'00101':Setting is prohibited '00110': Sub frame 1 channel length is 7 bits '00111': Sub frame 1 channel length is 8 bits ... ... '11110': Sub frame 1 channel length is 31 bits '11111': Sub frame 1 channel length is 32 bits Channel length can be set to 32 bits or less regardless of I2Sn_CNTREG:RHLL.
[20:16]	S1WDL	Sub Frame 1 Word Length word length of the channel constructing sub frame 1 (bit length of word) is set. 7 to 32 bits of word length are available but 1 to 6 bits are prohibited. S1WDL needs to be set to 'word length - 1' S1WDL needs to be set less than or equal to the value set in I2Sn_MCR0REG:S1CHL. Setting examples are shown below. '00000-00101':Setting is prohibited '00110': Sub frame 1 word length is 7 bits '00111': Sub frame 1 word length is 8 bits ... ... '11110': Sub frame 1 word length is 31 bits '11111': Sub frame 1 word length is 32 bits **Notes:** 1. If I2Sn_CNTREG:RHLL is '1', set word length to 16 bits or less. 2. If I2Sn_CNTREG:RHLL is '0', set word length to 32 bits or less. S1WDL is valid only in 2 sub frame construction (I2Sn_CNTREG:SBFN is '1') and is invalid in 1 sub frame construction (I2Sn_CNTREG:SBFN is '0').
[15]	read0	-

Table 34-5. Channel Control Register 0 (I2Sn_MCR0REG) bits

Bit Position	Bit Field Name	Bit Description
[14:10]	S0CHN	Sub Frame 0 Channel Numbers Number of channels of sub frame 0 is set up to 32 channels. S0CHN needs to be set to 'number of channel - 1'. Setting examples are shown below. '00000': Sub frame 0 becomes 1 channel construction '00001': Sub frame 0 becomes 2 channel construction '00010': Sub frame 0 becomes 3 channel construction ... ... '11110': Sub frame 0 becomes 31 channel construction '11111': Sub frame 0 becomes 32 channel construction
[9:5]	S0CHL	Sub Frame 0 Channel Length Channel length of the channel constructing sub frame 0 (bit length of channel) is set. 7 to 32 bits of channel length are available but 1 to 6 bits are prohibited. S0CHL needs to be set to 'channel length - 1'. Setting examples are shown below. '00000'-'00101':Setting is prohibited '00110': Sub frame 0 channel length is 7 bits '00111': Sub frame 0 channel length is 8 bits ... ... '11110': Sub frame 0 channel length is 31 bits '11111': Sub frame 0 channel length is 32 bits Channel length can be set to 32 bits or less regardless of I2Sn_CNTREG:RHLL

Table 34-5. Channel Control Register 0 (I2Sn_MCR0REG) bits

Bit Position	Bit Field Name	Bit Description
[4:0]	S0WDL	Sub Frame 1 Word Length word length of the channel constructing sub frame 0 (bit length of channel) is set. 7 to 32 bits of word length are available but 1 to 6 bits are prohibited. S0WDL needs to be set to 'word length - 1'. S0WDL needs to be set less than or equal to the value set in I2Sn_MCR0REG:S0CHL. Setting examples are shown below. '00000'-'00101':Setting is prohibited '00110': Sub frame 0 word length is 7 bits '00111': Sub frame 0 word length is 8 bits ... ... '11110': Sub frame 0 word length is 31 bits '11111': Sub frame 0 word length is 32 bits **Notes:** 1. If I2Sn_CNTREG:RHLL is '1', set word length to 16 bits or less. 2. If I2Sn_CNTREG:RHLL is '0', set word length to 32 bits or less.

34.2.5 Channel Control Register 1 (I2Sn_MCR1REG)

The Channel Control Register 1 (I2Sn_MCR1REG) controls enable and disable functions to each channel of sub frame 0.

Channel Control Register 1 (I2Sn_MCR1REG)

Figure 34-6. Channel Control Register 1 (I2Sn_MCR1REG)

I2Sn_MCR1REG																															
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**
S0CH[31]	S0CH[30]	S0CH[29]	S0CH[28]	S0CH[27]	S0CH[26]	S0CH[25]	S0CH[24]	S0CH[23]	S0CH[22]	S0CH[21]	S0CH[20]	S0CH[19]	S0CH[18]	S0CH[17]	S0CH[16]	S0CH[15]	S0CH[14]	S0CH[13]	S0CH[12]	S0CH[11]	S0CH[10]	S0CH[9]	S0CH[8]	S0CH[7]	S0CH[6]	S0CH[5]	S0CH[4]	S0CH[3]	S0CH[2]	S0CH[1]	S0CH[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-6. Channel Control Register 1 (I2Sn_MCR1REG) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	S0CH	Sub Frame 0 Channel Enable Each bit enables/disables corresponding channel of sub frame 0 (e.g. S0CH[0] bit controls 0th channel of sub frame 0, S0CH[31] bit controls 31st channel of sub frame 0). '0': The corresponding channel is disabled Transmission/reception is not performed to the disabled channel. '1': The corresponding channel is enabled Transmission/reception is performed to the enabled channel.

34.2.6 Channel Control Register 2 (I2Sn_MCR2REG)

The Channel Control Register 2 (I2Sn_MCR2REG) controls enable and disable functions to each channel of sub frame 1.

Channel Control Register 2 (I2Sn_MCR2REG)

Figure 34-7. Channel Control Register 2 (I2Sn_MCR2REG)

I2Sn_MCR2REG																															
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**
S1CH[31]	S1CH[30]	S1CH[29]	S1CH[28]	S1CH[27]	S1CH[26]	S1CH[25]	S1CH[24]	S1CH[23]	S1CH[22]	S1CH[21]	S1CH[20]	S1CH[19]	S1CH[18]	S1CH[17]	S1CH[16]	S1CH[15]	S1CH[14]	S1CH[13]	S1CH[12]	S1CH[11]	S1CH[10]	S1CH[9]	S1CH[8]	S1CH[7]	S1CH[6]	S1CH[5]	S1CH[4]	S1CH[3]	S1CH[2]	S1CH[1]	S1CH[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-7. Channel Control Register 2 (I2Sn_MCR2REG) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	S1CH	Sub Frame 1 Channel Enable Each bit enables/disables corresponding channel of sub frame 1 (e.g. S1CH[0] bit controls 0th channel of sub frame 1), S1CH[31] bit controls 31st channel of sub frame 1. When frame is 1 sub frame construction (I2Sn_CNTREG:SBFN is '0'), this is invalid. '0': The corresponding channel is disabled Transmission/ reception is not performed to the disabled channel. '1': The corresponding channel is enabled Transmission/reception is performed to the enabled channel.

34.2.7 Operation Control Register (I2Sn_OPRREG)

The Operation Control Register (I2Sn_OPRREG) controls receive and transmit operation.

Operation Control Register (I2Sn_OPRREG)

Figure 34-8. Operation Control Register (I2Sn_OPRREG)

I2Sn_OPRREG																															
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**
read0	read0	read0	read0	read0	read0	read0	RXENB	read0	read0	read0	read0	read0	read0	read0	TXENB	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	START
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-8. Operation Control Register (I2Sn_OPRREG) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	RXENB	Receive Enable Enable/disable functions of receiving operation is set. '0': Receiving operation is disabled Reception FIFO becomes empty with writing '0' to this bit. When RXENB is '0', the data received from serial reception bus is not written to reception FIFO. DMA reception channel stops during DMA transfer. '1': Receiving operation is enabled
[23:17]	read0	-
[16]	TXENB	Transmit Enable Enable/disable functions of transmitting operation is set. '0': Transmit operation is disabled Transmit FIFO becomes empty with writing '0' to this bit. When TXENB is '0', the data written to Transmission FIFO Data Registers from CPU or DMA is not written to transmission FIFO. DMA transmit channel stops during DMA transfer. '1': Transmit operation is enabled
[15:8]	read0	-
[7:1]	read0	-

Table 34-8. Operation Control Register (I2Sn_OPRREG) bits

Bit Position	Bit Field Name	Bit Description
[0]	START	I2S Enable I2S is enabled/disabled. '0': I2S is stopped and internal transmission/reception FIFO becomes empty by writing '0' to this bit '1': I2S is operable When START is '1', it is prohibited to rewrite I2Sn_CNTREG, I2Sn_MCR0REG, I2Sn_MCR1REG, and I2Sn_MCR2REG registers.

34.2.8 Software Reset Register (I2Sn_SRST)

This register is to control software reset.

Software Reset Register (I2Sn_SRST)

Figure 34-9. Software Reset Register (I2Sn_SRST)

I2Sn_SRST																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SRST
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-9. Software Reset Register (I2Sn_SRST) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	SRST	Software Reset Software reset is performed by writing '1'. I2Sn_STATUS register and each internal state machine enter initial state by software reset and transmission/reception FIFO becomes empty. There is no influence to registers other than I2Sn_STATUS, I2Sn_INTCNT, and I2Sn_DMAACT registers. When read value is '0' after writing '1', it indicates software reset is completed. '1' indicates software reset is in process.

34.2.9 Interrupt Control Register (I2Sn_INTCNT)

This register is to enable or disable interrupt function.

Interrupt Control Register (I2Sn_INTCNT)

Figure 34-10. Interrupt Control Register (I2Sn_INTCNT)

I2Sn_INTCNT																															
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**
read0	TXUD1M	TBERM	FERRM	TXUD0M	TXOVM	TXFDM	TXFIM	read0	read0	RBERM	RXUDM	RXOVM	EOPM	RXFDM	RXFIM	read0	read0	read0	read0	TFTH[3]	TFTH[2]	TFTH[1]	TFTH[0]	read0	read0	RPTMR[1]	RPTMR[0]	RFTH[3]	RFTH[2]	RFTH[1]	RFTH[0]
Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-10. Interrupt Control Register (I2Sn_INTCNT) bits

Bit Position	Bit Field Name	Bit Description
[31]	read0	-
[30]	TXUD1M	Tx FIFO Underflow Interrupt Mask This is transmission FIFO underflow interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:TXUDR1 is not masked '1': Interrupt to CPU by I2Sn_STATUS:TXUDR1 is masked
[29]	TBERM	Tx Block Size Error Interrupt Mask This is interrupt mask bit of block size error of transmission channel. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:TBERR is not masked '1': Interrupt to CPU by I2Sn_STATUS:TBERR is masked
[28]	FERRM	Frame Error Interrupt Mask This is frame error interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:FERR is not masked '1': Interrupt to CPU by I2Sn_STATUS:FERR is masked
[27]	TXUD0M	Tx FIFO Underflow Interrupt Mask This is transmission FIFO underflow interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:TXUDR0 is not masked '1': Interrupt to CPU by I2Sn_STATUS:TXUDR0 is masked

Table 34-10. Interrupt Control Register (I2Sn_INTCNT) bits

Bit Position	Bit Field Name	Bit Description
[26]	TXOVM	Tx FIFO Overflow Interrupt Mask This is transmission FIFO overflow interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:TXOVR is not masked '1': Interrupt to CPU by I2Sn_STATUS:TXOVR is masked
[25]	TXFDM	Tx DMA Mask This is transmission DMA request mask register bit. It becomes '1' by software reset. '0': DMA transfer is requested when empty space in transmission FIFO is more than threshold value '1': DMA transfer is not requested even when empty space in transmission FIFO is more than threshold value
[24]	TXFIM	Tx FIFO Interrupt Mask This is transmission FIFO interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:TXFI is not masked '1': Interrupt to CPU by I2Sn_STATUS:TXFI is masked
[23:22]	read0	-
[21]	RBERM	Rx Block Size Error Interrupt Mask This is interrupt mask bit of reception channel block size error. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:RBERR is not masked '1': Interrupt to CPU by I2Sn_STATUS:RBERR is masked
[20]	RXUDM	Rx FIFO Underflow Interrupt Mask This is reception FIFO underflow interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:RXUDR is not masked '1': Interrupt to CPU by I2Sn_STATUS:RXUDR is masked
[19]	RXOVM	Rx FIFO Overflow Interrupt Mask This is interrupt mask bit of reception FIFO overflow. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:RXOVR is not masked '1': Interrupt to CPU by I2Sn_STATUS:RXOVR is masked
[18]	EOPM	EOPI Interrupt Mask This is interrupt mask bit by EOPI of status register. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:EOPI is not masked '1': Interrupt to CPU by I2Sn_STATUS:EOPI is masked

Table 34-10. Interrupt Control Register (I2Sn_INTCNT) bits

Bit Position	Bit Field Name	Bit Description
[17]	RXFDM	Rx FIFO DMA Mask This is reception DMA request mask bit. It becomes '1' by software reset. '0': DMA transfer is requested when reception data written to reception FIFO is more than threshold value, previous Rx DMA block transfer has completed and there is no Rx block size error. '1': DMA transfer is not requested
[16]	RXFIM	Rx FIFO Interrupt Mask This is reception FIFO interrupt mask bit. It becomes '1' by software reset. '0': Interrupt to CPU by I2Sn_STATUS:RXFI is not masked '1': Interrupt to CPU by I2Sn_STATUS:RXFI is masked
[15:12]	read0	-
[11:8]	TFTH	Tx FIFO Threshold Threshold value of transmission FIFO is set. Empty space of transmission FIFO is more than threshold value and I2Sn_INTCNT:Interrupt to CPU occurs. Empty space of transmission FIFO is more than threshold value, previous Tx DMA block transfer has completed, there is no Tx block size error I2Sn_INTCNT:TXFDM is '0': DMA is requested to DMAC.
[7:6]	read0	-
[5:4]	RPTMR	Rx Completion Timer This is packet reception completion timer setting bit which sets timeout value of the internal reception completion timer. Reception FIFO is not empty and number of its data is smaller than or equal to threshold value:the timer always counts up. Reception FIFO is empty or the data value is more than threshold value:the timer is cleared. When the timer times out, I2Sn_STATUS:EOPI bit is set to '1'. The timer becomes '00' by software reset. '00': 0 (the timer is not in operation) '01': 54000 CLK_PERI4_PD2 cycles '10': 108000 CLK_PERI4_PD2 cycles '11': 216000 CLK_PERI4_PD2 cycles

Table 34-10. Interrupt Control Register (I2Sn_INTCNT) bits

Bit Position	Bit Field Name	Bit Description
[3:0]	RFTH	Rx FIFO Threshold Threshold value of reception FIFO is set. Number of receive words in reception FIFO is more than thresh-old value and I2Sn_INTCNT:Interrupt to CPU occurs. Number of receive words in reception FIFO is more than threshold value, previous Rx DMA block transfer has completed, there is no Rx block size errorand I2Sn_INTCNT:RXFDM is '0': DMA is requested to DMAC.

34.2.10 Status Register (I2Sn_STATUS)

This register gives status information.

Status Register (I2Sn_STATUS)

Figure 34-11. Status Register (I2Sn_STATUS)

I2Sn_STATUS																															
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**
TBERR	RBERR	FERR	TXUDR1	TXUDR0	TXOVR	RXUDR	RXOVR	read0	read0	read0	read0	EOPI	BSY	TXFI	RXFI	TXNUM[7]	TXNUM[6]	TXNUM[5]	TXNUM[4]	TXNUM[3]	TXNUM[2]	TXNUM[1]	TXNUM[0]	RXNUM[7]	RXNUM[6]	RXNUM[5]	RXNUM[4]	RXNUM[3]	RXNUM[2]	RXNUM[1]	RXNUM[0]
Rp	Rp	RpWp1	RpWp1	RpWp1	RpWp1	RpWp1	RpWp1	Rp0	Rp0	Rp0	Rp0	RpWp1	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-11. Status Register (I2Sn_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[31]	TBERR	Tx Block Size Error When I2Sn_DMAACT:TDMACT is '1' and block transfer through DMA transmission channel is more than I2Sn_INTCNT:TFTH + 1, this bit is set to 1 and the DMA transmission channel is stopped. When TBERR is '1' and I2Sn_INTCNT:TBERM is '0', interrupt to CPU occurs. This bit becomes '0' by software reset.
[30]	RBERR	Rx Block Size Error When I2Sn_DMAACT:RDMACT is '1' and block transfer through DMA reception channel is more than I2Sn_INTCNT:RFTH + 1, this bit is set to 1 and the DMA reception channel is stopped. When RBERR is '1' and I2Sn_INTCNT:RBERM is '0', interrupt to CPU occurs. This bit becomes '0' by software reset.

Table 34-11. Status Register (I2Sn_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[29]	FERR	Frame Error Occurrence of frame error is indicated. This bit is set to '1' in the following cases: ■ Frame synchronous signal can not be received with the set frame rate in the free-running mode (I2Sn_CNTREG:FRUN = '0') and the slave mode (I2Sn_CNTREG:MSMD = '0'). ■ The next frame synchronous signal is received during frame transmission/reception in the slave mode (I2Sn_CNTREG:MSMD), not free-running mode (I2Sn_CNTREG:FRUN = 1). When FERR is '1' and I2Sn_INTCNT:FERRM is '0', interrupt to CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset.
[28]	TXUDR1	Tx FIFO Underflow Error When transmission FIFO underflows at the start of frame, the value is set to '1'. When TXUDR1 is '1' and I2Sn_INTCNT:TXUD1M is '0', interrupt to the CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset. **Note:** When the transmission FIFO underflows at the start of frame, there will be no more attempts to read the transmission FIFO during the rest of the frame, so if the transmission FIFO is not written after it has underflown, I2Sn_STATUS:TXUDR0 is not set to '1'.
[27]	TXUDR0	Tx FIFO Underflow Error When transmission FIFO underflows during frame transmission (from 2nd bit word to the last frame of the word), the value is set to '1'. When TXUDR0 is '1' and I2Sn_INTCNT:TXUD0M is '0', interrupt to the CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset. **Note:** When the transmission FIFO underflows during frame transmission and is still empty at the start of the next frame, I2Sn_STATUS:TXUDR1 is set to '1' too.

Table 34-11. Status Register (I2Sn_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[26]	TXOVR	Tx FIFO Overflow Error When transmission FIFO overflows, the value is set to '1' indicating transmission data is written in the condition that transmission FIFO is full. The value '1' indicates 1 word or more of transmission data is ignored. When TXOVR is '1' and I2Sn_INTCNT:TXOVM is '0', interrupt to CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset.
[25]	RXUDR	Rx FIFO Underflow Error When reception FIFO underflows, the value is set to '1' indicating read access is carried out to reception FIFO in the condition that reception FIFO is empty. When RXUDR is '1' and I2Sn_INTCNT:RXUDM is '0', interrupt to the CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset. **Note:** This flag is not set when the DAP controller reads the Reception FIFO Data Registers while the reception FIFO is empty.
[24]	RXOVR	Rx FIFO Overflow Error When reception FIFO overflows, the value is set to '1' indicating reception is carried out in the condition that reception FIFO is full. The value '1' indicates 1 word or more of reception data is ignored. When RXOVR is '1' and I2Sn_INTCNT:RXOVM is '0', interrupt to CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset.
[23:20]	read0	-

Table 34-11. Status Register (I2Sn_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[19]	EOPI	Interrupt Flag for Rx Timer This is an interrupt flag that is triggered when an internal reception timer times out. The reception timer is enabled when following conditions are met at the same time: I2Sn_CNTREG:RXDIS is set to '0'. I2Sn_OPRREG:START bit is set to '1' and I2Sn_OPRREG:RXENB = '1'. After the reset, operation starts with the 1st word reception. The count value is automatically cleared if reception FIFO data is more than threshold or it becomes empty. When the reception FIFO is not empty and the number of data is less than or equal to the threshold, the reception timer is incremented with each CLK_PERI4_PD2 cycle. EOPI is set to '1' when reception FIFO is non-empty and reception timer count value reaches timeout set by I2Sn_INTCNT:RPTMR. When EOPI is '1' and I2Sn_INTCNT:EOPM is '0', interrupt to CPU occurs. Writing '1' from CPU clears the value to '0'. This bit becomes '0' by software reset.
[18]	BSY	Serial Tx Busy Serial transmission control part is in busy state. This bit is not affected by software reset. '0': Serial transmission control part is in idle state '1': Serial transmission control part is in busy state
[17]	TXFI	Tx FIFO Empty When number of empty space of transmission FIFO is greater than the threshold set in I2Sn_INTCNT:TFTH, this bit is set to '1'. This bit is '1' and I2Sn_INTCNT:interrupt to CPU occurs. When number of empty slot of transmission FIFO becomes smaller or equal to the threshold by writing to Transmission FIFO Data Registers from CPU or DMAC, this bit is cleared automatically to '0'. The value also become '0' when I2Sn_OPRREG:START bit is '0' or I2Sn_OPRREG:TXENB bit is '0'. If software reset is performed at I2Sn_OPRREG:START bit = '1' and I2Sn_OPRREG:TXENB bit = '1', the value becomes '0' during software reset then changes to '1' after the process.

Table 34-11. Status Register (I2Sn_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[16]	RXFI	Rx FIFO Full When number of reception FIFO data becomes more than the threshold set in I2Sn_INTCNT:RFTH, this bit is set to '1'. This bit is '1' and I2Sn_INTCNT:interrupt to CPU occurs. When number of data in reception FIFO becomes smaller or equal to the threshold by reading Reception FIFO DATA Registers from CPU or DMAC, this bit is automatically cleared to '0'. This bit becomes '0' by software reset.
[15:8]	TXNUM	Number of Tx FIFO Data The number of data in transmission FIFO is indicated. This field is incremented by write access to Transmission FIFO Data Registers and decremented by serial word transmission. Max. value of 66 can be displayed in the simultaneous transfer mode and value of 132 in the transmission only mode. This field becomes '00000000' by software reset.
[7:0]	RXNUM	Number of Data in Rx FIFO The number of data in reception FIFO is indicated. This field is incremented by word reception from serial bus and decremented by read access to Reception FIFO Data Registers. Maximum value of 66 can be displayed in the simultaneous transfer mode and value of 132 in the reception mode. This field becomes '00000000' by software reset.

34.2.11 DMA Activate Register (I2Sn_DMAACT)

This register is to enable or disable DMA control function.

DMA Activate Register (I2Sn_DMAACT)

Figure 34-12. DMA Activate Register (I2Sn_DMAACT)

I2Sn_DMAACT																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TDMACT	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RDMACT
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-12. DMA Activate Register (I2Sn_DMAACT) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	TDMACT	Tx DMA Control The DMA transmission channel is activated. After transmission channel starts, software should write '1' to TDMACT for transmit DMA channel to be active. When TDMACT is '0', transfer request of transmission channel block is not sent to DMAC. Writing '0' from CPU clears the value to '0'. This bit becomes '0' by software reset. '0': DMA transmission channel is disabled '1': DMA transmission channel is enabled Clearing TDMACT also clears write transmission request.
[15:8]	read0	-
[7:1]	read0	-

Table 34-12. DMA Activate Register (I2Sn_DMAACT) bits

Bit Position	Bit Field Name	Bit Description
[0]	RDMACT	Rx DMA Control The DMA reception channel is activated. After reception channel starts, software should write '1' to RDMACT for receive DMA channel to be active. When RDMACT is '0', transfer request of reception channel block is not sent to DMAC. Writing '0' from CPU clears the value to '0'. This bit becomes '0' by software reset. '0': DMA reception channel is disabled '1': DMA reception channel is enabled Clearing RDMACT also clears reception transfer request.

34.2.12 Debug Register (I2Sn_DEBUG)

This register is to enable or disable debug function.

Debug Register (I2Sn_DEBUG)

Figure 34-13. Debug Register (I2Sn_DEBUG)

I2Sn_DEBUG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DBGE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-13. Debug Register (I2Sn_DEBUG) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	DBGE	Debug Enable (DBGE) This bit is used to enable/disable debug mode for I2S. '0': Debug mode disabled '1': Debug mode enabled This bit takes effect only in master mode (i.e. I2Sn_CNTREG:MSMD = '1'). When DBGE is set to '1' and the processor is in debug state and I2S is working as a master, then the serial interface is halted by stopping the activity on the SCK output. The activity on the serial clock resumes either when the processor leaves debug state or DBGE is set to '0'. For the definition of debug state, refer to section 11.8 of the Arm Cortex-R4 Technical Reference Manual.

34.2.13 Module ID Register (I2Sn_MIDREG)

This register implements unique module identification number. Refer to the device datasheet for the module identification number of I2S module in the device.

Module ID Register (I2Sn_MIDREG)

Figure 34-14. Module ID Register (I2Sn_MIDREG)

I2Sn_MIDREG																															
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**
MID[31]	MID[30]	MID[29]	MID[28]	MID[27]	MID[26]	MID[25]	MID[24]	MID[23]	MID[22]	MID[21]	MID[20]	MID[19]	MID[18]	MID[17]	MID[16]	MID[15]	MID[14]	MID[13]	MID[12]	MID[11]	MID[10]	MID[9]	MID[8]	MID[7]	MID[6]	MID[5]	MID[4]	MID[3]	MID[2]	MID[1]	MID[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34-14. Module ID Register (I2Sn_MIDREG) bits

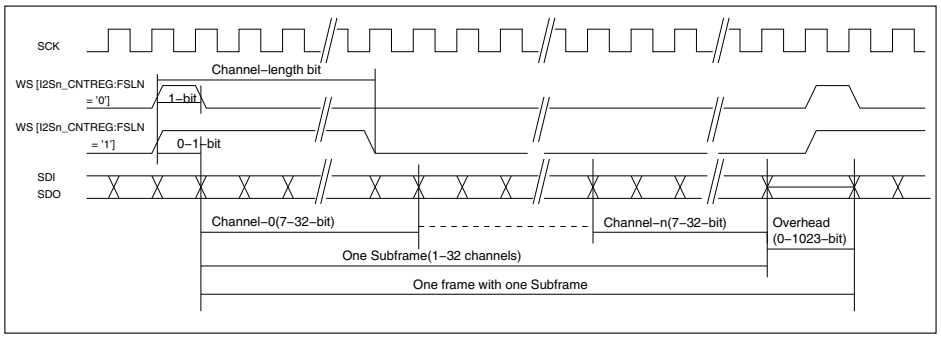
Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID This read-only register gives the unique module identification number of I2S module. The unique Module ID number identifies the version of the I2S module used in the MCU. Refer to the device specific datasheet for the module identification number of its I2S.

34.3 I2S frame construction

I2S supports frame format of multiple channel construction. Frame can be configured to 1 or 2 sub frames. Number of each frame's channel and word length can be set individually.

1 sub frame construction

Figure 34-15. 1 sub frame construction

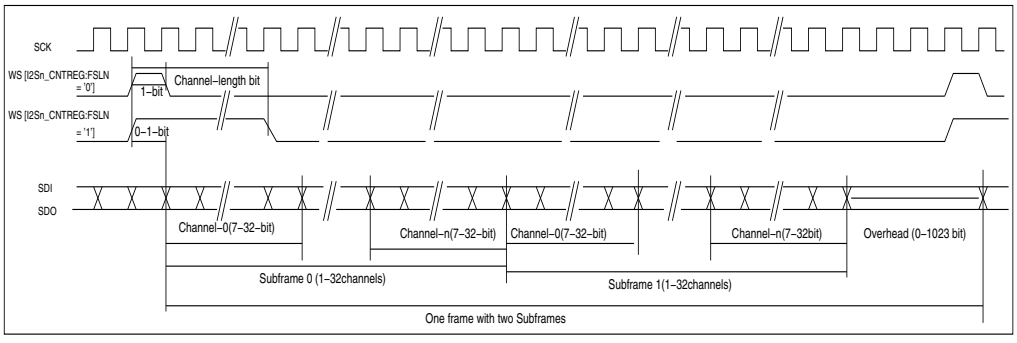


Description

- When I2Sn_CNTREG:SBFN bit is '0', frame becomes 1 sub frame composite
- Number of channels of 1-sub frame is determined by I2Sn_MCR0REG:S0CHN
- Up to 32 channels can be set
- Each channel bit length (word length) is determined by I2Sn_MCR0REG:S0WDL
- Sub frame channel starts from 0th. Each channel is set to valid/invalid with the corresponding bit of I2Sn_MCR1REG register. Transmission/reception of data is not performed to invalid channel
- Dummy bit can be inserted behind sub frame by setting I2Sn_CNTREG:OVHD. 0 to 1023 bits are insertable
- Polarity of WS is set with I2Sn_CNTREG:FSPL
- Pulse width of WS can be set to 1 bit or 1 channel length by setting I2Sn_CNTREG:FSLN
- Frame sync phase of WS can be set to '0' or '1' clock through I2Sn_CNTREG:FSPH
- In this construction, settings of I2Sn_MCR0REG:S1CHN and I2Sn_MCR2REG:S1WDL are ignored

2 sub frame construction

Figure 34-16. 2 sub frame construction



Description

- When I2Sn_CNTREG:SBFN bit is '1', frame becomes 2 sub frame composition
- Set number of channel of sub frame 0 to I2Sn_MCR0REG:S0CHN, and set number of sub frame 1 channel to I2Sn_MCR0REG:S1CHN. Up to 32 channels can be set

- Channel bit length (word length) of sub frame 0 is determined by I2Sn_MCR0REG:S0WDL. For sub frame 1, they are determined by I2Sn_MCR0REG:S1WDL
- Sub frame channel starts from 0th. Each channel of sub frame 0 is set to valid/invalid with the corresponding bit of I2Sn_MCR1REG register, and corresponding bit of I2Sn_MCR2REG register for sub frame 1
- Transmission/reception of data is not performed to invalid channel
- Dummy bit can be inserted behind sub frame 1 by setting I2Sn_CNTREG:OVHD. 0 to 1023 bits are insertable
- Polarity of WS is set to I2Sn_CNTREG:FSPL bit
- Pulse width of WS can be set to 1 bit or 1 channel length by setting I2Sn_CNTREG:FSLN bit, channel length setting of 1 channel is determined by the channel length of sub frame '0'
- Frame sync phase of WS can be set to '0' or '1' clock through I2Sn_CNTREG:FSPH bit

34.4 I2S configuration and operation modes

Transmission only mode

Table 34-15. Transmission only mode

Transfer setting	Operation	Master mode (I2Sn_CNTREG:MSMD = '1')	Slave mode (I2Sn_CNTREG:MSMD = '0')
Transmission only I2Sn_CNTREG: TXDIS = '0' I2Sn_CNTREG: RXDIS = '1'	Start	**Free-running mode (I2Sn_CNTREG:FRUN = '1'):** After I2Sn_OPRREG:START bit becomes ’1’ and I2Sn_OPRREG:TXENB bit is ’1’, frame synchronous signal starts to output when transmission FIFO is not empty. From the second time, frame synchronous signal is output with the frame rate determined by the register setting. If transmission FIFO is empty, empty frame is output at the same time of frame synchronous signal output. Serial data of the empty frame can be set to ’0’ or ’1’ by the register setting. **Burst mode (I2Sn_CNTREG:FRUN = '0'):** If transmission FIFO is not empty, I2Sn_OPRREG:START bit is ’1’ and I2Sn_OPRREG:TXENB bit is ’1’, frame synchronous signal is output. After completion of one frame output, transmission FIFO status is always confirmed. If transmission FIFO is not empty, frame synchronous signal is output to perform frame transmission.	**Free-running mode (I2Sn_CNTREG:FRUN = '1'):** The frame synchronous signal is input at the frame rate determined by the register setting. If transmission FIFO is empty when the frame synchronous signal is input when I2Sn_OPRREG:START bit is ’1’ and I2Sn_OPRREG:TXENB bit is ’1’, empty frame is output. Serial data of the empty frame can be set to '0' or '1' by the register setting. **Burst mode (I2Sn_CNTREG:FRUN = '0'):** When I2Sn_OPRREG:START bit is ’1’ and I2Sn_OPRREG:TXENB bit is ’1’, one frame is output every time the frame synchronous signal is input. When transmission FIFO is empty at the time of frame synchronous signal input, empty frame is output.

Table 34-15. Transmission only mode

Transfer setting	Operation	Master mode (I2Sn_CNTREG:MSMD = '1')	Slave mode (I2Sn_CNTREG:MSMD = '0')
	Stop	At the time of stop, transmission FIFO becomes empty when there's no data transfer from internal memory to I2S transmission FIFO. To maintain I2Sn_OPRREG:START bit to **'1':** I2Sn_OPRREG:TXENB = '1': When '1' is written to I2Sn_OPRREG:TXENB, synchronous signal is output in the freerunning mode. When transmission FIFO becomes empty, empty frame is output. In burst mode, frame synchronous signal is not output, and empty frame bits are output to serial data bus. I2Sn_OPRREG:TXENB = '0': When '0' is written to I2Sn_OPRREG:TXENB, transmission FIFO becomes empty. In the free-running mode, frame synchronous signal continues outputting and serial bus becomes high impedance state. In the burst mode, frame synchronous signal is not output and serial data bus becomes high impedance state. **To make I2Sn_OPRREG:START bit '0':** When '0' is written to I2Sn_OPRREG:START bit, then transmission FIFO becomes empty. Clock supply to the serial control part is stopped regardless of I2Sn_OPRREG:TXENB setting. Serial Output Clock and Frame synchronous signal output is stopped. Serial data bus becomes high impedance state.	**To maintain I2Sn_OPRREG:START bit to '1':** I2Sn_OPRREG:TXENB = '1': Empty frame data is output to serial bus. I2Sn_OPRREG:TXENB = '0': When '0' is written to I2Sn_OPRREG:TXENB, transmission FIFO becomes empty, and data present in the transmission FIFO at the time '0' was written to I2Sn_OPRREG:TXENB is not transmitted. Writing to transmission FIFO and detection of the frame synchronous signal are stopped. Serial data bus becomes high impedance state. To make I2Sn_OPRREG:START bit '0': When '0' is written to I2Sn_OPRREG:START bit, transmission FIFO becomes empty. Writing to transmission FIFO and detection of frame synchronous signal are stopped regardless of I2Sn_OPRREG:TXENB setting and serial bus becomes high impedance state.
	Abnormality	When reading from transmission FIFO occurs while it is empty, empty frame is output. For the setting conditions of I2Sn_STATUS:TXUDR0 and I2Sn_STATUS:TXUDR1, refer to their bit descriptions. When writing to transmission FIFO occurs while it is full, set I2Sn_STATUS:TXOVR to '1'.	When reading from transmission FIFO occurs while it is empty, empty frame is output. For the setting conditions of I2Sn_STATUS:TXUDR0 and I2Sn_STATUS:TXUDR1, refer to their bit descriptions. However I2Sn_STATUS:TXUDR0/1 are not set to '1' for the 1st output frame after the bits become I2Sn_OPRREG:START = '1' and I2Sn_OPRREG:TXENB = '1'. When writing to transmission FIFO occurs while it is full, I2Sn_STATUS:TXOVR is set to '1'. If the frame synchronous signal is not input with the defined frame rate in the free-running mode, I2Sn_STATUS:FERR is set to '1'. If the next frame synchronous signal is input before completing 1 frame transmission in the burst mode, I2Sn_STATUS:FERR is set to '1'.

Reception only mode

Table 34-16. Reception only mode

Transfer Setting	Operation	Master mode (I2Sn_CNTREG:MSMD = '1')	Slave mode (I2Sn_CNTREG:MSMD = '0')
Reception only I2Sn_CNTREG: TXDIS = '1' I2Sn_CNTREG: RXDIS = '0'	Start	**Free-running mode (I2Sn_CNTREG:FRUN = '1'):** Frame synchronous signal starts to output after I2Sn_OPRREG:START bit becomes '1' and I2Sn_OPRREG:RXENB bit is '1' when reception FIFO is not full. From the second time, frame synchronous signal with the frame rate determined by the register setting is output. **Burst mode (I2Sn_CNTREG:FRUN = '0'):** When I2Sn_OPRREG:START bit is ’1’ and I2Sn_OPRREG:RXENB bit is ’1’, frame synchronous signal is output to receive frame if reception FIFO is not full. If the FIFO is full, the signal is not output.	**Free-running mode (I2Sn_CNTREG:FRUN = '1'):** When I2Sn_OPRREG:START bit is '1' and I2Sn_OPRREG:RXENB bit is '1', input frame synchronous signal with the frame rate determined by the register setting. Frame should be received every time the signal is input. **Burst mode (I2Sn_CNTREG:FRUN = '0'):** When I2Sn_OPRREG:START bit is ’1’ and I2Sn_OPRREG:RXENB bit is ’1’, frame reception is performed every time frame synchronous signal is input. The signal is input with less speed than the frame rate in the free-running mode.
	Stop	At the time of stop, frame is not imported from serial bus even though reception FIFO is empty. To maintain I2Sn_OPRREG:START bit to **'1':** When ’0’ is written to I2Sn_OPRREG:RXENB, reception FIFO becomes empty. Although frame synchronous signal is kept outputting in the free-running mode, frame is not received. In the burst mode, frame is not received and the signal is not output. **To make I2Sn_OPRREG:START bit '0':** When ’0’ is written to I2Sn_OPRREG:START, reception FIFO becomes empty. Clock supply to the serial control part stops regardless of I2Sn_OPRREG:RXENB setting, and SCK supply to the external part is stopped as well.	To maintain I2Sn_OPRREG:START bit to **'1':** Reception FIFO becomes empty by writing '0' to I2Sn_OPRREG:RXENB. The input frame synchronous signal is ignored, and frames are not received. To make I2Sn_OPRREG:START bit '0': When ’0’ is written to the I2Sn_OPRREG:START bit, the rececption FIFO becomes empty. The input frame synchronous signal is ignored regardless of I2Sn_OPRREG:RXENB setting, and frames are not received.
	Abnormality	When writing to reception FIFO occurs while it is full, I2Sn_STATUS:RXOVR is set to ’1’. I2Sn_STATUS:RXUDR bit is set to ’1’ when read access to reception FIFO occurs while it is empty.	When writing to reception FIFO occurs while it is full, I2Sn_STATUS:RXOVR is set to ’1’. When read access to reception FIFO occurs while it is empty, I2Sn_STATUS:RXUDR is set to ’1’. Free-running mode: If frame synchronous signal is not input with the frame rate defined by the register setting, I2Sn_STATUS:FERR bit is set to ’1’.

Simultaneous transfer mode

Table 34-17. Simultaneous transfer mode

Transfer setting	Operation	Master mode (I2Sn_CNTREG:MSMD = '1')	Slave mode (I2Sn_CNTREG:MSMD = '0')
Simultaneous Transfer I2Sn_CNTREG: TXDIS = ‘0’ I2Sn_CNTREG: RXDIS = ‘0’	Start	**Free-running mode (I2Sn_CNTREG:FRUN = '1'):** I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’1’, and I2Sn_OPRREG:RXENB = ’0’: The same operation as transmission only mode. I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’0’, and I2Sn_OPRREG:RXENB = ’1’: The same operation as reception only mode. I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’1’, and I2Sn_OPRREG:RXENB = ’1’: Frame synchronous signal is output from the state that transmission FIFO is not empty and reception FIFO is not full. Then frame synchronous signal is output with the frame rate defined by the register setting. At the same time, empty frame is output if transmission FIFO is empty. Empty frame’s serial data can be set to ’0’ or ’1’ by register setting. Every time frame synchronous signal is output, frame is received. **Burst mode (I2Sn_CNTREG:FRUN = '0'):** I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’1’, and I2Sn_OPRREG:RXENB = ’0’: The same operation as transmission only mode. I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’0’, and I2Sn_OPRREG:RXENB = ’1’: The same operation as reception only mode. I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’1’, and I2Sn_OPRREG:RXENB = ’1’: Frame synchronous signal is output from the state that transmission FIFO is not empty and reception FIFO is not full. After completion of one frame output or at idle state, transmission/reception FIFO status is always confirmed. If transmission FIFO is not empty and reception FIFO is not full, frame synchronous signal is output to perform frame transmission/reception.	**Free-running mode (I2Sn_CNTREG:FRUN = '1'):** I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’1’, and I2Sn_OPRREG:RXENB = ’0’: The same operation as transmission only mode. I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’0’, and I2Sn_OPRREG:RXENB = ’1’: The same operation as reception only mode. I2Sn_OPRREG:START = ’1’, I2Sn_OPRREG:TXENB = ’1’, and I2Sn_OPRREG:RXENB = ’1’: Frame synchronous signal is input with the frame rate defined by the register setting. At the same time, empty frame is output if transmission FIFO is empty. The serial data can be set to ’0’ or ’1’ by the register setting I2Sn_CNTREG:MSKB. Every time frame synchronous signal is input, frame is received. **Burst mode (I2Sn_CNTREG:FRUN = '0'):** Every time frame synchronous signal is input when I2Sn_OPRREG:START bit is ’1’, transmission and reception for one frame is performed. When the frame synchronous signal is input, empty frame is output if transmission FIFO is empty.

Table 34-17. Simultaneous transfer mode

Transfer setting	Operation	Master mode (I2Sn_CNTREG:MSMD = '1')	Slave mode (I2Sn_CNTREG:MSMD = '0')
	Stop	Stop operation has following states: Transmission stop: Transmission FIFO becomes empty when data is not transferred from internal memory to I2S transmission FIFO. Reception stop: Data does not need to be transferred from I2S reception FIFO to internal memory. **To maintain I2Sn_OPRREG:START bit to '1':** In free-running mode frame synchronous signal is output. In the burst mode, when transmission FIFO becomes empty, frame synchronous output is stopped. Transmission stop: I2Sn_OPRREG:TXENB = '1': Empty frame bit is output when transmission FIFO becomes empty. I2Sn_OPRREG:TXENB = '0': Transmission FIFO becomes empty and transmission serial data bus becomes high impedance. Writing to transmission FIFO stops. Reception stop: When ’0’ is written to I2Sn_OPRREG:RXENB, reception FIFO becomes empty and frame reception operation stops. **To make I2Sn_OPRREG:START bit '0'**: When ’0’ is written to I2Sn_OPRREG:START, transmission/reception FIFO becomes empty. The clock supply to the internal serial control part stops regardless of I2Sn_OPRREG:TXENB and I2Sn_OPRREG:RXENB status. SCK output to the external part and frame synchronous signal output is also stopped.	**To maintain I2Sn_OPRREG:START bit to '1'**: Transmission stop: When I2Sn_OPRREG:TXENB = ’1’, empty frame bits are output after transmission FIFO becomes empty. When ’0’ is written to I2Sn_OPRREG:TXENB, transmisison FIFO becomes empty and transmission serial data bus becomes high impedance. Data present in the transmission FIFO at the time ’0’ was written to I2Sn_OPRREG:TXENB is not transmitted. While I2Sn_OPRREG:TXENB = ’0’, data is not written to transmission FIFO. Reception stop: When ’0’ is written to I2Sn_OPRREG:RXENB, reception FIFO becomes empty and frame reception operation stops. **To make I2Sn_OPRREG:START bit '0'**: When ’0’ is written to I2Sn_OPRREG:START, transmission/reception FIFO becomes empty. Transmission/reception is stopped regardless of I2Sn_OPRREG:TXENB and I2Sn_OPRREG:RXENB status.

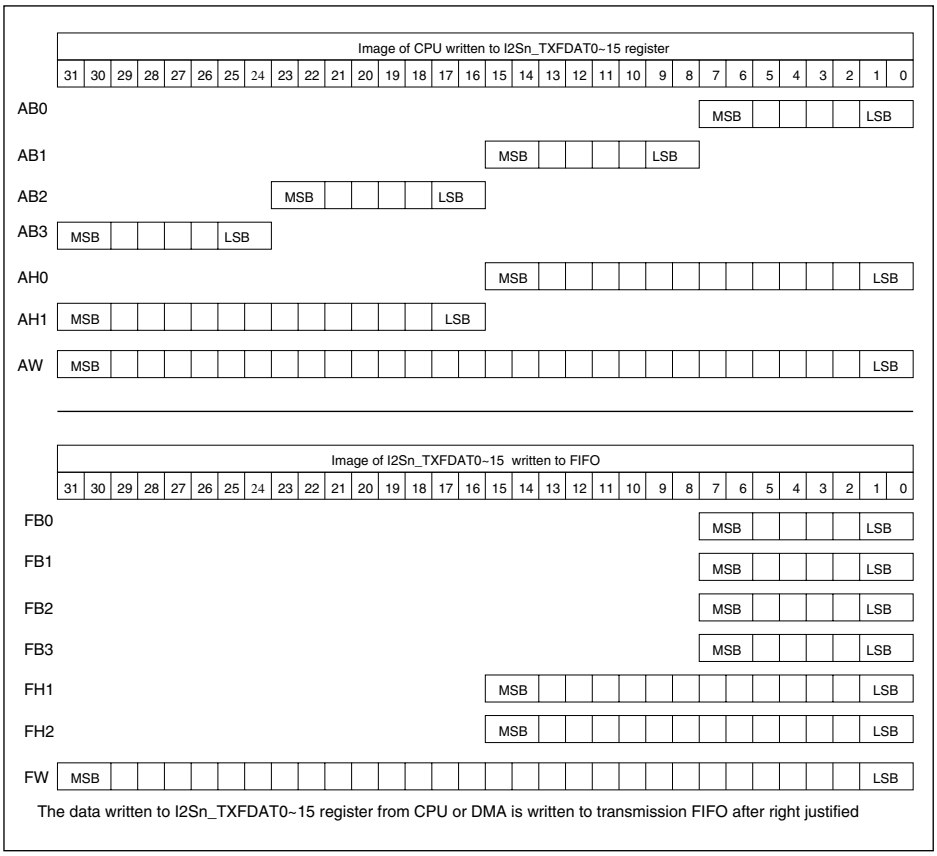
Table 34-17. Simultaneous transfer mode

Transfer setting	Operation	Master mode (I2Sn_CNTREG:MSMD = '1')	Slave mode (I2Sn_CNTREG:MSMD = '0')
	Abnormality	When reading from transmission FIFO occurs while it is empty, empty frame bit is output. For the setting conditions of I2Sn_STATUS:TXUDR0 and I2Sn_STATUS:TXUDR1, refer to their bit descriptions. When writing to transmission FIFO occurs while it is full, I2Sn_STATUS:TXOVR is set to ’1’. When read access occurs to reception FIFO while it is empty, I2Sn_STATUS:RXUDR is set to ’1’. When writing to reception FIFO occurs while it is full, I2Sn_STATUS:RXOVR is set to ’1’.	When reading from transmission FIFO occurs while it is empty, empty frame bit is output. For the setting conditions of I2Sn_STATUS:TXUDR0 and I2Sn_STATUS:TXUDR1, refer to their bit descriptions. When writing to transmission FIFO occurs while it is full, I2Sn_STATUS:TXOVR is set to ’1’. When read access occurs to reception FIFO while it is empty, I2Sn_STATUS:RXUDR is set to ’1’. When writing to reception FIFO occurs while it is full, I2Sn_STATUS:RXOVR is set to ’1’. If the frame synchronous signal is not input with the defined frame rate in the free-running mode, I2Sn_STATUS:FERR is set to ’1’. If the following frame synchronous signal is input before completing one frame transmission in the burst mode, I2Sn_STATUS:FERR is set to ’1’.

34.5 Bit alignment

Transmission word alignment

Figure 34-17. Transmission word line chart



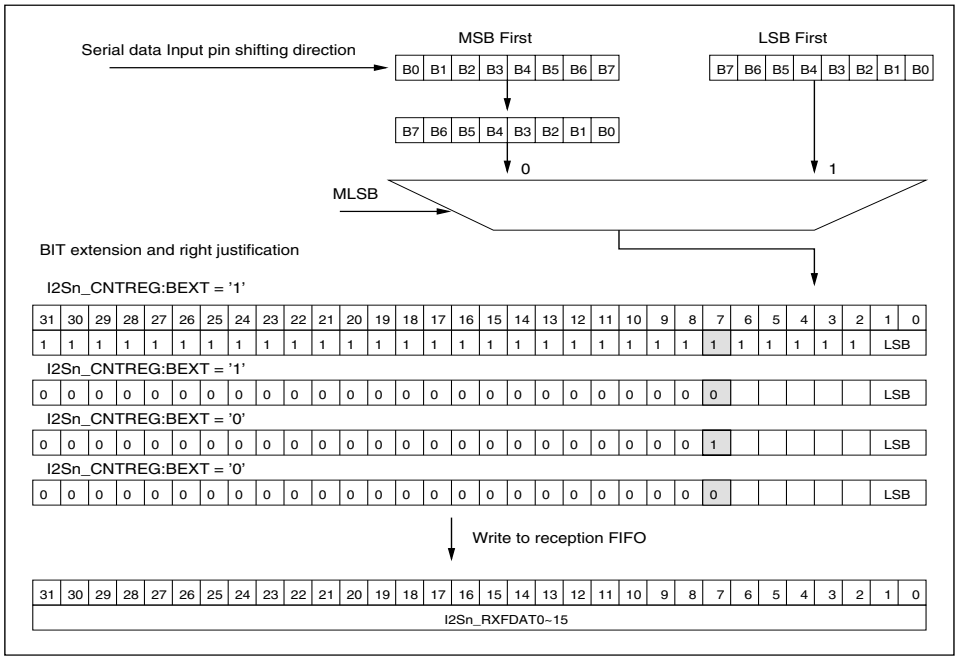
When transmission is performed with serial bus, word is sent MSB first when I2Sn_CNTREG:MSLB is '0' and LSB first when the value is '1'. When channel length (set to I2Sn_MCR0REG:S0CHL and I2Sn_MCR0REG:S1CHL) is longer than the word length (set to I2Sn_MCR0REG:S0WDL and I2Sn_MCR0REG:S1WDL), remaining bits in the channel become I2Sn_CNTREG:MSKB. Setting the channel length to shorter than the word length is prohibited.

Notes:

- AB0, AB1, AB2, AB3, AH0, AH1, and AW on the above chart indicate byte 0, byte 1, byte 2, byte 3, half word 0, half word 1, and word at write accessing to I2Sn_TXFDAT0~15 on AHB bus
- Each FB0, FB1, FB2, FB3, FH0, FH1, and FW indicate AB0, AB1, AB2, AB3, AH0, AH1, and AW at writing to transmission FIFO after they are right justified

Reception word alignment

Figure 34-18. Reception word line chart



This chart shows word line example of when word length is 8.

The word received from serial bus is always written to reception FIFO after being right justified.

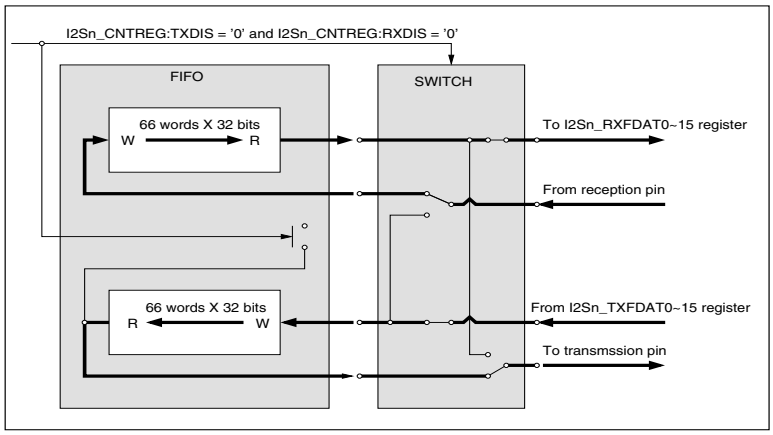
Therefore, read access should be performed from AHB bus to I2Sn_RXFDAT0~15 in order to read as follows:

Word length:

- 8 or less: byte 0
- 9- 6: half word 0
- 17-32: all words

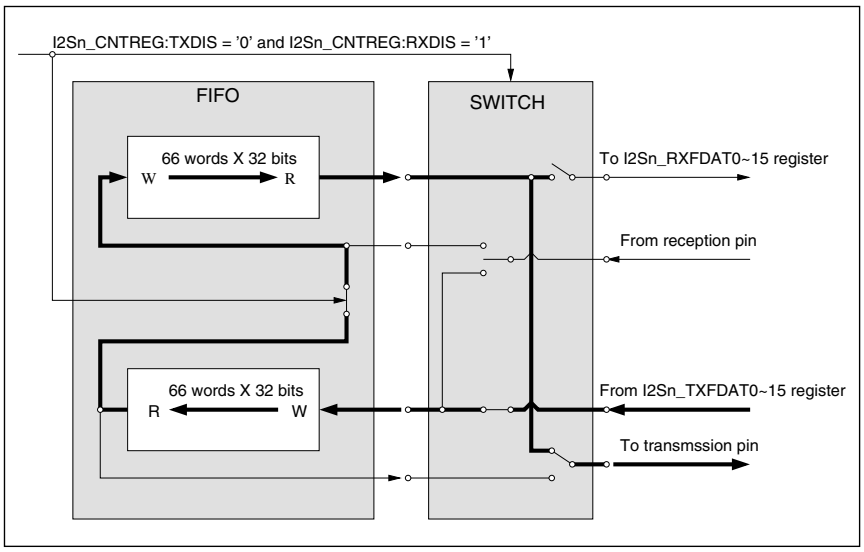
34.6 FIFO construction

Figure 34-19. Simultaneous transfer mode data flow



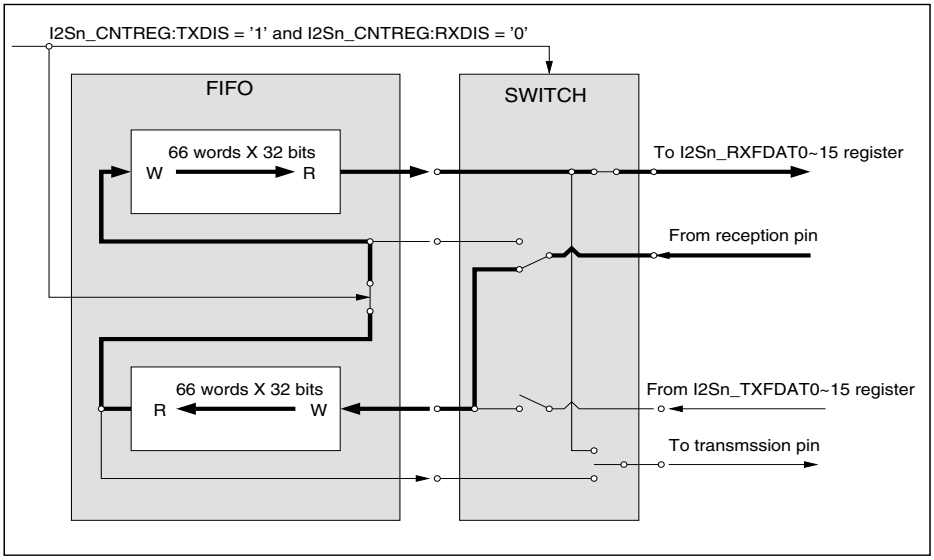
When I2Sn_CNTREG:TXDIS = '0' and I2Sn_CNTREG:RXDIS = '0', the mode is set to simultaneous transfer mode which operates with 66-word x 32-bit transmission and reception FIFOs.

Figure 34-20. Transmission only mode data flow



When I2Sn_CNTREG:TXDIS = '0' and I2Sn_CNTREG:RXDIS = '1', the mode is set to transmission only mode which operates with a 132-word x 32-bit transmission FIFO, and reception is not performed.

Figure 34-21. Reception only mode data flow



When I2Sn_CNTREG:TXDIS = '1' and I2Sn_CNTREG:RXDIS = '0', the mode is set to reception only mode which operates with a 132-word x 32-bit reception FIFO, and transmission is not performed.

34.7 Caution summary

- I2Sn_MCR0REG:S0WDL, I2Sn_MCR0REG:S0CHL 1 to 6 bits are prohibited
- I2Sn_MCR0REG:S1WDL, I2Sn_MCR0REG:S1CHL 1 to 6 bits are prohibited
- When channel length (set to I2Sn_MCR0REG:S0CHL and I2Sn_MCR0REG:S1CHL) is longer than the word length (set to I2Sn_MCR0REG:S0WDL and I2Sn_MCR0REG:S1WDL), remaining bits in the channel become I2Sn_CNTREG:MSKB. Setting the channel length to shorter than the word length is prohibited
- Pulse width of one channel (I2Sn_CNTREG:FSLN = '1') is prohibited when frame length is set to one channel by I2Sn_MCR0REG:S0CHN = 0 and I2Sn_CNTREG:SBFN = '0'
- Rewrite to I2Sn_CNTREG, I2Sn_MCR0REG, I2Sn_MCR1REG, and I2Sn_MCR2REG is prohibited after I2Sn_OPRREG:START is set
- Rewrite to I2Sn_CNTREG, I2Sn_MCR0REG, I2Sn_MCR1REG, and I2Sn_MCR2REG is prohibited while DBGE is set to ’1’ and the processor is in debug state

35. GPIO Module

This chapter explains the function and operation of the General Purpose Input/Output (GPIO) module.

35.1 Outline of the GPIO module

This section describes the features and the block diagram of the GPIO module.

Features of GPIO module

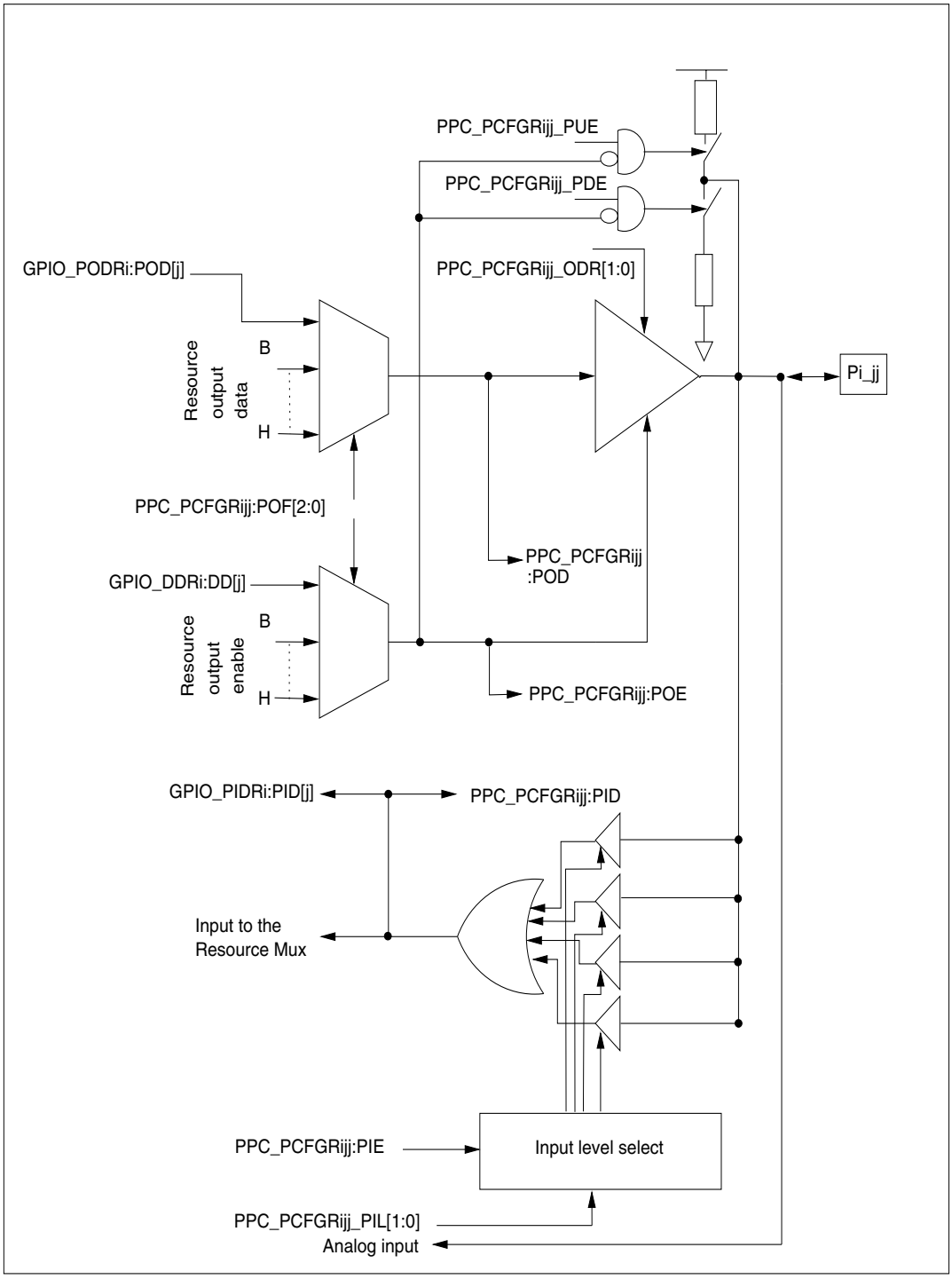
The General Purpose Input/Output (GPIO) module provides 512 GPIO channels. Each GPIO channel has a corresponding port pin, which is controlled by the channel only if 'General Purpose Port' is selected by the POF[2:0] bits in the corresponding Pin Configuration Register (PPC_PCFGRijj). The features of the GPIO module are:

- It accommodates eight GPIO ports
- Each GPIO port accommodates 64 GPIO channels
- Each GPIO channel is in turn mapped to a corresponding port pin
- The GPIO module accommodates eight 64-bit registers (GPIO_DDRi, GPIO_DDSRi, GPIO_DDCRi, GPIO_PODRi, GPIO_POSRi, GPIO_POCRi, GPIO_PIDRi, GPIO_PPERi), all of which are associated with a single GPIO port
- Each register is repeated eight times to support the eight GPIO ports
- The eight GPIO ports support 512 GPIO channels and in turn 512 port pins
- Each GPIO module register can be accessed by 8, 16, 32 or 64-bit bus accesses
- All GPIO module registers are readable
- All GPIO module registers (except GPIO_PPERi and GPIO_PIDRi) are bitwise write-protected
- Port Pin Enable Registers (GPIO_PPERi) are used for the bitwise enabling of write accesses to the corresponding GPIO module registers. Moreover, they enable read/write access to the corresponding Pin Configuration registers in the Port Pin Configuration module
- GPIO_PPERi can be written only once in privileged mode; any further writes are blocked and signaled to the Bus Error Correction Unit (BECU) as a Peripheral Protection Unit (PPU) error

Block diagram of the GPIO module I/O port

Figure 35-1 shows the block diagram of the GPIO module and Port Pin Configuration module. The Data Direction Register (GPIO_DDRi) and Port Output Data Register (GPIO_PODRi), together with the Pin Configuration Register (PPC_PCFGRijj), control each port pin. For detailed a description of the PPC_PCFGRijj register refer to 18. Port Pin Configuration. The Port Input Data Register (GPIO_PIDRi) contains the value detected on the corresponding port pin.

Figure 35-1. Block diagram of GPIO module I/O port



35.2 GPIO module registers

This section describes the registers of the GPIO module in detail.

The suffix 'i' in the register name indicates that the register corresponds to the 64-bit port i within the GPIO module.

The numbering of the bits within a 64-bit port is done by 'j' (0..63) or by the two-digit number 'jj' (00..63).

Registers of the GPIO module

The following registers are available for each 64-bit port within the GPIO module:

- Data Direction Register (GPIO_DDRi)
- Data Direction Set Register (GPIO_DDSRi)
- Data Direction Clear Register (GPIO_DDCRi)
- Port Output Data Register (GPIO_PODRi)
- Port Output Set Register (GPIO_POSRi)
- Port Output Clear Register (GPIO_POCRi)
- Port Input Data Register (GPIO_PIDRi)
- Port Pin Enable Register (GPIO_PPERi)

Memory layout of the GPIO module registers

The GPIO module uses 1 KB of address space for mapping its own Control and Status Registers. All GPIO module registers are shown in Table 35-1. Data in the registers is available using 8-, 16-, 32- and 64-bit accesses.

Table 35-1. Memory layout of the GPIO module registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	GPIO_POSR0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000008	GPIO_POCR0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000010	GPIO_DDSR0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000018	GPIO_DDCR0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000020	GPIO_POSR1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000028	GPIO_POCR1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000030	GPIO_DDSR1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000038	GPIO_DDCR1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000040	GPIO_POSR2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000048	GPIO_POCR2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							

Table 35-1. Memory layout of the GPIO module registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000050	GPIO_DDSR2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000058	GPIO_DDCR2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000060	GPIO_POSR3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000068	GPIO_POCR3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000070	GPIO_DDSR3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000078	GPIO_DDCR3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000080	GPIO_POSR4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000088	GPIO_POCR4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000090	GPIO_DDSR4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000098	GPIO_DDCR4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000A0	GPIO_POSR5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000A8	GPIO_POCR5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000B0	GPIO_DDSR5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000B8	GPIO_DDCR5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000C0	GPIO_POSR6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000C8	GPIO_POCR6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000D0	GPIO_DDSR6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000D8	GPIO_DDCR6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000E0	GPIO_POSR7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							

Table 35-1. Memory layout of the GPIO module registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x000000E8	GPIO_POCR7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000F0	GPIO_DDSR7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000000F8	GPIO_DDCR7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000100 - 000001F8	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000200	GPIO_PODR0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000208	GPIO_DDR0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000210	GPIO_PODR1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000218	GPIO_DDR1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000220	GPIO_PODR2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000228	GPIO_DDR2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000230	GPIO_PODR3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000238	GPIO_DDR3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000240	GPIO_PODR4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000248	GPIO_DDR4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000250	GPIO_PODR5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000258	GPIO_DDR5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000260	GPIO_PODR6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000268	GPIO_DDR6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000270	GPIO_PODR7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							

Table 35-1. Memory layout of the GPIO module registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000278	GPIO_DDR7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000280 - 000002F8	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000300	GPIO_PIDR0 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000308	GPIO_PIDR1 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000310	GPIO_PIDR2 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000318	GPIO_PIDR3 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000320	GPIO_PIDR4 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000328	GPIO_PIDR5 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000330	GPIO_PIDR6 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000338	GPIO_PIDR7 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000340 - 00000378	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000380	GPIO_PPER0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000388	GPIO_PPER1 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000390	GPIO_PPER2 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x00000398	GPIO_PPER3 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000003A0	GPIO_PPER4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000003A8	GPIO_PPER5 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000003B0	GPIO_PPER6 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							
0x000003B8	GPIO_PPER7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000							

Numbering of the GPIO channels

Table 35-1 describes the association between the GPIO ports, channels, module registers and port pins. For illustration purposes, only one register, the Data Direction Register (GPIO_DDRi) is shown below. All other registers follow the same scheme.

Table 35-2. Numbering of GPIO channels

GPIO port number	GPIO channel number	Associated GPIO module register	Port pin number
GPIO port 0	GPIO0_00 to GPIO0_63	GPIO_DDR0 bits [0 to 63]	P0_00 to P0_63
GPIO port 1	GPIO1_00 to GPIO1_63	GPIO_DDR1 bits [0 to 63]	P1_00 to P1_63
GPIO port 2	GPIO2_00 to GPIO2_63	GPIO_DDR2 bits [0 to 63]	P2_00 to P2_63
GPIO port 3	GPIO3_00 to GPIO3_63	GPIO_DDR3 bits [0 to 63]	P3_00 to P3_63
GPIO port 4	GPIO4_00 to GPIO4_63	GPIO_DDR4 bits [0 to 63]	P4_00 to P4_63
GPIO port 5	GPIO5_00 to GPIO5_63	GPIO_DDR5 bits [0 to 63]	P5_00 to P5_63
GPIO port 6	GPIO6_00 to GPIO6_63	GPIO_DDR6 bits [0 to 63]	P6_00 to P6_63
GPIO port 7	GPIO7_00 to GPIO7_63	GPIO_DDR7 bits [0 to 63]	P7_00 to P7_63

35.2.1 Data Direction Register (GPIO_DDRi)

The Data Direction Register controls the direction of one GPIO port. It contains 64 bits to control the direction of 64 GPIO channels. The port pin of the device, which corresponds to a particular GPIO channel, is controlled by the GPIO channel only if ‘General Purpose Port’ is selected by the POF[2:0] bits in the corresponding Pin Configuration Register (PPC_PCFGRijj). The GPIO_DDRi register can be written to directly or it can be set and cleared using the GPIO_DDSRi and GPIO_DDCRi registers respectively.

Data Direction Register (GPIO_DDRi)

Figure 35-2. Data Direction Register (GPIO_DDRi)

GPIO_DDR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
DD[63]	DD[62]	DD[61]	DD[60]	DD[59]	DD[58]	DD[57]	DD[56]	DD[55]	DD[54]	DD[53]	DD[52]	DD[51]	DD[50]	DD[49]	DD[48]	DD[47]	DD[46]	DD[45]	DD[44]	DD[43]	DD[42]	DD[41]	DD[40]	DD[39]	DD[38]	DD[37]	DD[36]	DD[35]	DD[34]	DD[33]	DD[32]
RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
DD[31]	DD[30]	DD[29]	DD[28]	DD[27]	DD[26]	DD[25]	DD[24]	DD[23]	DD[22]	DD[21]	DD[20]	DD[19]	DD[18]	DD[17]	DD[16]	DD[15]	DD[14]	DD[13]	DD[12]	DD[11]	DD[10]	DD[9]	DD[8]	DD[7]	DD[6]	DD[5]	DD[4]	DD[3]	DD[2]	DD[1]	DD[0]
RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-3. Data Direction Register (GPIO_DDRi) bits

Bit Position	Bit Field Name	Bit Description
[63:0]	DD	Data Direction This register controls the direction for one GPIO port. '0': Output is disabled for the port pin corresponding to the GPIO channel '1': Output is enabled for the port pin corresponding to the GPIO channel

Note:

The GPIO module provides eight Data Direction Registers (GPIO_DDR0 to GPIO_DDR7) to control the output enable for 512 GPIO channels and hence 512 port pins.

35.2.2 Data Direction Set Register (GPIO_DDSRi)

The Data Direction Set Register is used to set all corresponding bits in the GPIO_DDRi register. Reading the GPIO_DDSRi register always returns zero.

Data Direction Set Register (GPIO_DDSRi)

Figure 35-3. Data Direction Set Register (GPIO_DDSRi)

GPIO_DDSR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
DDS[63]	DDS[62]	DDS[61]	DDS[60]	DDS[59]	DDS[58]	DDS[57]	DDS[56]	DDS[55]	DDS[54]	DDS[53]	DDS[52]	DDS[51]	DDS[50]	DDS[49]	DDS[48]	DDS[47]	DDS[46]	DDS[45]	DDS[44]	DDS[43]	DDS[42]	DDS[41]	DDS[40]	DDS[39]	DDS[38]	DDS[37]	DDS[36]	DDS[35]	DDS[34]	DDS[33]	DDS[32]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
DDS[31]	DDS[30]	DDS[29]	DDS[28]	DDS[27]	DDS[26]	DDS[25]	DDS[24]	DDS[23]	DDS[22]	DDS[21]	DDS[20]	DDS[19]	DDS[18]	DDS[17]	DDS[16]	DDS[15]	DDS[14]	DDS[13]	DDS[12]	DDS[11]	DDS[10]	DDS[9]	DDS[8]	DDS[7]	DDS[6]	DDS[5]	DDS[4]	DDS[3]	DDS[2]	DDS[1]	DDS[0]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-4. Data Direction Set Register (GPIO_DDSRi)

Bit Position	Bit Field Name	Bit Description
[63:0]	DDS	Data Direction Set This register is used to set the corresponding bits in the Data Direction Register (GPIO_DDRi). '0': No effect '1': Sets the corresponding bit of the Data Direction Register (GPIO_DDRi) Reading this register always returns '0'.

Note: The GPIO module provides eight Data Direction Set Registers (GPIO_DDSR0 to GPIO_DDSR7) corresponding to the eight GPIO_DDRi registers.

35.2.3 Data Direction Clear Register (GPIO_DDCRi)

Data Direction Clear Register is used to clear all corresponding bits in GPIO_DDRi register. Reading the GPIO_DDCRi register always returns zero.

Data Direction Clear Register (GPIO_DDCRi)

Figure 35-4. Data Direction Clear Register (GPIO_DDCRi)

GPIO_DDCR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
DDC[63]	DDC[62]	DDC[61]	DDC[60]	DDC[59]	DDC[58]	DDC[57]	DDC[56]	DDC[55]	DDC[54]	DDC[53]	DDC[52]	DDC[51]	DDC[50]	DDC[49]	DDC[48]	DDC[47]	DDC[46]	DDC[45]	DDC[44]	DDC[43]	DDC[42]	DDC[41]	DDC[40]	DDC[39]	DDC[38]	DDC[37]	DDC[36]	DDC[35]	DDC[34]	DDC[33]	DDC[32]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
DDC[31]	DDC[30]	DDC[29]	DDC[28]	DDC[27]	DDC[26]	DDC[25]	DDC[24]	DDC[23]	DDC[22]	DDC[21]	DDC[20]	DDC[19]	DDC[18]	DDC[17]	DDC[16]	DDC[15]	DDC[14]	DDC[13]	DDC[12]	DDC[11]	DDC[10]	DDC[9]	DDC[8]	DDC[7]	DDC[6]	DDC[5]	DDC[4]	DDC[3]	DDC[2]	DDC[1]	DDC[0]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-5. Data Direction Clear Register (GPIO_DDCRi) bits

Bit Position	Bit Field Name	Bit Description
[63:0]	DDC	Data Direction Clear This register is used to clear the corresponding bits in the Data Direction Register (GPIO_DDRi). '0': No effect '1': Clears the corresponding bit in the Data Direction Register (GPIO_DDRi) Reading this register always returns '0'.

Note: The GPIO module provides eight Data Direction Clear Registers (GPIO_DDCR0 to GPIO_DDCR7) corresponding to the eight GPIO_DDRi registers.

35.2.4 Port Output Data Register (GPIO_PODRi)

The Port Output Data Register controls the output value of the GPIO channel if the corresponding bit in the GPIO_DDRi register is set to '1' (i.e. when the GPIO channel is in output mode). This register can be written to directly or can be set and cleared using the GPIO_POSRi and GPIO_POCRi registers respectively.

Port Output Data Register (GPIO_PODRi)

Figure 35-5. Port Output Data Register (GPIO_PODRi)

GPIO_PODR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
POD[63]	POD[62]	POD[61]	POD[60]	POD[59]	POD[58]	POD[57]	POD[56]	POD[55]	POD[54]	POD[53]	POD[52]	POD[51]	POD[50]	POD[49]	POD[48]	POD[47]	POD[46]	POD[45]	POD[44]	POD[43]	POD[42]	POD[41]	POD[40]	POD[39]	POD[38]	POD[37]	POD[36]	POD[35]	POD[34]	POD[33]	POD[32]
RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
POD[31]	POD[30]	POD[29]	POD[28]	POD[27]	POD[26]	POD[25]	POD[24]	POD[23]	POD[22]	POD[21]	POD[20]	POD[19]	POD[18]	POD[17]	POD[16]	POD[15]	POD[14]	POD[13]	POD[12]	POD[11]	POD[10]	POD[9]	POD[8]	POD[7]	POD[6]	POD[5]	POD[4]	POD[3]	POD[2]	POD[1]	POD[0]
RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp	RWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-6. Port Output Data Register (GPIO_PODRi)

Bit Position	Bit Field Name	Bit Description
[63:0]	POD	Port Output Data Bits in this register control the output value on the corresponding GPIO channel. '0': Logic '0' is driven to the corresponding GPIO channel output if the corresponding bit in the GPIO_DDRi register is set to '1' '1': Logic '1' is driven to the corresponding GPIO channel output if the corresponding bit in the GPIO_DDRi register is set to '1'

Note: The GPIO module provides eight Port Output Data Registers (GPIO_PODR0 to GPIO_PODR7) to control the output values of 512 GPIO channels and hence 512 port pins.

35.2.5 Port Output Set Register (GPIO_POSRi)

The Port Output Set Register is used to set all corresponding bits in GPIO_PODRi register. Reading the GPIO_POSRi register always returns zero.

Port Output Set Register (GPIO_POSRi)

Figure 35-6. Port Output Set Register (GPIO_POSRi)

GPIO_POSR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
POS[63]	POS[62]	POS[61]	POS[60]	POS[59]	POS[58]	POS[57]	POS[56]	POS[55]	POS[54]	POS[53]	POS[52]	POS[51]	POS[50]	POS[49]	POS[48]	POS[47]	POS[46]	POS[45]	POS[44]	POS[43]	POS[42]	POS[41]	POS[40]	POS[39]	POS[38]	POS[37]	POS[36]	POS[35]	POS[34]	POS[33]	POS[32]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
POS[31]	POS[30]	POS[29]	POS[28]	POS[27]	POS[26]	POS[25]	POS[24]	POS[23]	POS[22]	POS[21]	POS[20]	POS[19]	POS[18]	POS[17]	POS[16]	POS[15]	POS[14]	POS[13]	POS[12]	POS[11]	POS[10]	POS[9]	POS[8]	POS[7]	POS[6]	POS[5]	POS[4]	POS[3]	POS[2]	POS[1]	POS[0]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-7. Port Output Set Register (GPIO_POSRi) bits

Bit Position	Bit Field Name	Bit Description
[63:0]	POS	Port Output Set This register is used to set the corresponding bits in the Port Output Data Register (GPIO_PODRi). '0': No effect '1': Sets the corresponding bit of the GPIO_PODRi register Reading this register always returns '0'.

Note: The GPIO module provides eight Port Output Set Registers (GPIO_POSR0 to GPIO_POSR7) corresponding to the eight GPIO_PODRi registers.

35.2.6 Port Output Clear Register (GPIO_POCRi)

The Port Output Clear Register is used to clear all corresponding bits in the GPIO_PODRi register. Reading the GPIO_POCRi register always returns zero.

Port Output Clear Register (GPIO_POCRi)

Figure 35-7. Port Output Clear Register (GPIO_POCRi)

GPIO_POCR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
POC[63]	POC[62]	POC[61]	POC[60]	POC[59]	POC[58]	POC[57]	POC[56]	POC[55]	POC[54]	POC[53]	POC[52]	POC[51]	POC[50]	POC[49]	POC[48]	POC[47]	POC[46]	POC[45]	POC[44]	POC[43]	POC[42]	POC[41]	POC[40]	POC[39]	POC[38]	POC[37]	POC[36]	POC[35]	POC[34]	POC[33]	POC[32]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
POC[31]	POC[30]	POC[29]	POC[28]	POC[27]	POC[26]	POC[25]	POC[24]	POC[23]	POC[22]	POC[21]	POC[20]	POC[19]	POC[18]	POC[17]	POC[16]	POC[15]	POC[14]	POC[13]	POC[12]	POC[11]	POC[10]	POC[9]	POC[8]	POC[7]	POC[6]	POC[5]	POC[4]	POC[3]	POC[2]	POC[1]	POC[0]
R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1	R0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-8. Port Output Clear Register (GPIO_POCRi)

Bit Position	Bit Field Name	Bit Description
[63:0]	POC	Port Output Clear This register is used to clear the corresponding bits in the Port Output Data Register (GPIO_PODRi). '0': No effect '1': Clears the corresponding bit in the GPIO_PODRi register Reading this register always returns '0'.

Note: The GPIO module provides eight Port Output Clear Registers (GPIO_POCR0 to GPIO_POCR7) corresponding to the eight GPIO_PODRi registers.

35.2.7 Port Input Data Register (GPIO_PIDRi)

The Port Input Data Register provides the status at the input of the GPIO channels. Each bit in the GPIO_PIDRi register indicates the input data of the corresponding port pin (independent of direction setting in GPIO_DDRi) detected at the input buffer, which is selected by the PPC_PCFGRijj:PIL[1:0] bits if PPC_PCFGRijj:PIE is set to '1'. If PPC_PCFGRijj:PIE is set to '0', then the corresponding GPIO_PIDRi bit returns X.

Port Input Data Register (GPIO_PIDRi)

Figure 35-8. Port Input Data Register (GPIO_PIDRi)

GPIO_PIDR0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
PID[63]	PID[62]	PID[61]	PID[60]	PID[59]	PID[58]	PID[57]	PID[56]	PID[55]	PID[54]	PID[53]	PID[52]	PID[51]	PID[50]	PID[49]	PID[48]	PID[47]	PID[46]	PID[45]	PID[44]	PID[43]	PID[42]	PID[41]	PID[40]	PID[39]	PID[38]	PID[37]	PID[36]	PID[35]	PID[34]	PID[33]	PID[32]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
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**
PID[31]	PID[30]	PID[29]	PID[28]	PID[27]	PID[26]	PID[25]	PID[24]	PID[23]	PID[22]	PID[21]	PID[20]	PID[19]	PID[18]	PID[17]	PID[16]	PID[15]	PID[14]	PID[13]	PID[12]	PID[11]	PID[10]	PID[9]	PID[8]	PID[7]	PID[6]	PID[5]	PID[4]	PID[3]	PID[2]	PID[1]	PID[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 35-9. Port Input Data Register (GPIO_PIDRi) bits

Bit Position	Bit Field Name	Bit Description
[63:0]	PID	Port Input Data Each read-only bit of the GPIO_PIDRi register indicates the input data of the corresponding port pin detected at the input buffer selected by PPC_PCFGRijj:PIL[1:0].If PPC_PCFGRijj:PIE = '0', then the corresponding GPIO_PIDRi bit returns X. '0': logic '0' is present at the corresponding port pin '1': logic '1' is present at the corresponding port pin Writing to this register is blocked and signaled as a PPU error.

Note: The GPIO module provides eight Port Input Data Registers (GPIO_PIDR0 to GPIO_PIDR7) to support 512 GPIO channels and hence 512 port pins.

35.2.8 Port Pin Enable Register (GPIO_PPERi)

The Port Pin Enable Register provides bitwise write-protection to the GPIO module registers as well as read- and write-protection to the port pin configuration registers corresponding to the respective GPIO channels.

Port Pin Enable Register (GPIO_PPERi)

Figure 35-9. Port Pin Enable Register (GPIO_PPERi)

GPIO_PPER0																															
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**
PPE[63]	PPE[62]	PPE[61]	PPE[60]	PPE[59]	PPE[58]	PPE[57]	PPE[56]	PPE[55]	PPE[54]	PPE[53]	PPE[52]	PPE[51]	PPE[50]	PPE[49]	PPE[48]	PPE[47]	PPE[46]	PPE[45]	PPE[44]	PPE[43]	PPE[42]	PPE[41]	PPE[40]	PPE[39]	PPE[38]	PPE[37]	PPE[36]	PPE[35]	PPE[34]	PPE[33]	PPE[32]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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**
PPE[31]	PPE[30]	PPE[29]	PPE[28]	PPE[27]	PPE[26]	PPE[25]	PPE[24]	PPE[23]	PPE[22]	PPE[21]	PPE[20]	PPE[19]	PPE[18]	PPE[17]	PPE[16]	PPE[15]	PPE[14]	PPE[13]	PPE[12]	PPE[11]	PPE[10]	PPE[9]	PPE[8]	PPE[7]	PPE[6]	PPE[5]	PPE[4]	PPE[3]	PPE[2]	PPE[1]	PPE[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 35-10. Port Pin Enable Register (GPIO_PPERi) bits

Bit Position	Bit Field Name	Bit Description
[63:0]	PPE	Port Pin Enable Each bit of GPIO_PPERi controls the write access to the corresponding bits of the GPIO module register, and read and write access to the corresponding Pin Configuration Registers (PPC_PCFGRijj). This register can be written once in privileged mode; further writes are blocked and signaled as a PPU error. Any write access in user mode is blocked and signaled as a PPU error. '0': Indicates that the registers of the corresponding port pin are not accessible, and the bit of the corresponding GPIO channel cannot be written to '1': Indicates that the registers of the corresponding port pin are accessible, and the bit of corresponding GPIO channel can be written to

Notes:

1. The GPIO module provides eight Port Pin Enable Registers (GPIO_PPER0 to GPIO_PPER7) to support 512 GPIO channels and hence 512 port pins.
2. Each byte of the GPIO_PPERi register can be written to only once in privileged mode. Separate protection exists for each individual byte within the GPIO_PPERi register.

35.3 Operation of the GPIO module

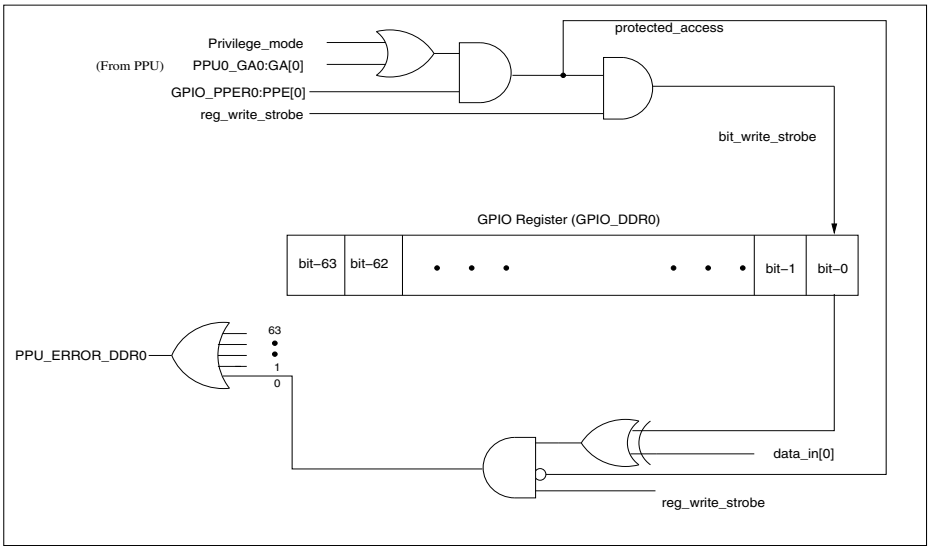
This chapter describes the protection mechanism used to write-protect each bit in the GPIO module registers.

Bitwise write-protection

All GPIO module registers (except GPIO_PPERi and GPIO_PIDRi) are bitwise write-protected. Figure 35-10 describes bitwise write-protection for single bit of a write-protected register. PPU error generation is illustrated for the GPIO module register GPIO_DDR0.

- In privileged mode, each bit in a GPIO module write-protected register can be written only if the corresponding bit in the Port Pin Enable Register (GPIO_PPERi:PPE[j]) is set to '1'
- In user mode, each bit in a GPIO module write-protected register can be written if the corresponding PPU GPIO Access Attribute Register bit in the PPU (PPU0_GAk:GA[l] where k = 2*i + int(j/32) and l = j mod 32) is set to '1', and the corresponding bit in the Port Pin Enable Register (GPIO_PPERi:PPE[j]) is also set to '1'
- The individual write strobe for each bit is generated by logically ANDing the ‘reg_write_strobe’ signal with the ‘protected_access’ signal (as shown in Figure 35-10)
- A PPU error is generated if the ‘bit_write_strobe’ signal is generated but the ‘protected_access’ signal is not asserted, and if the bit value being written is different than its current value

Figure 35-10. Bitwise write protection logic



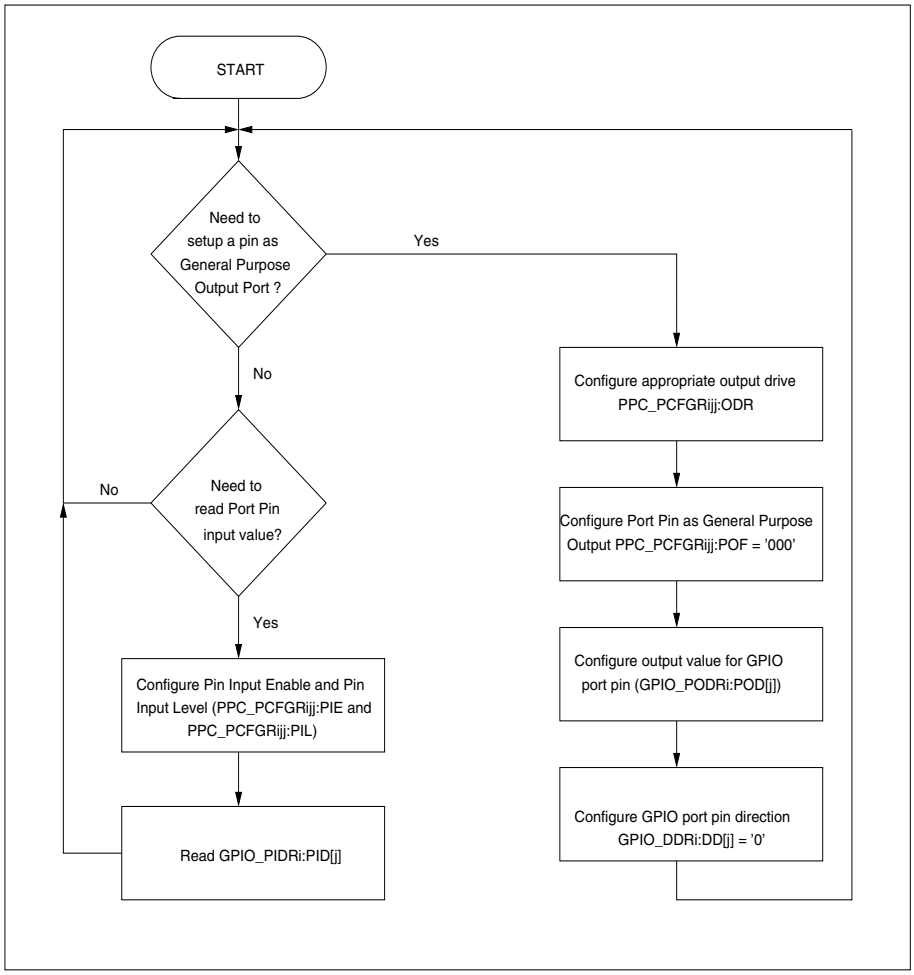
35.4 GPIO flowchart

This section describes the flowchart of the GPIO.

Steps in setting up the GPIO

The following flowchart displays the sequence of steps that must be followed to correctly set the direction (i.e. input or output) of the GPIO port.

Figure 35-11. Steps in setting up GPIO



36. 8/10-Bit Analog to Digital Converter

This chapter explains the functions and operations of the 8/10-bit A/D Converter.

36.1 Outline of A/D Converter

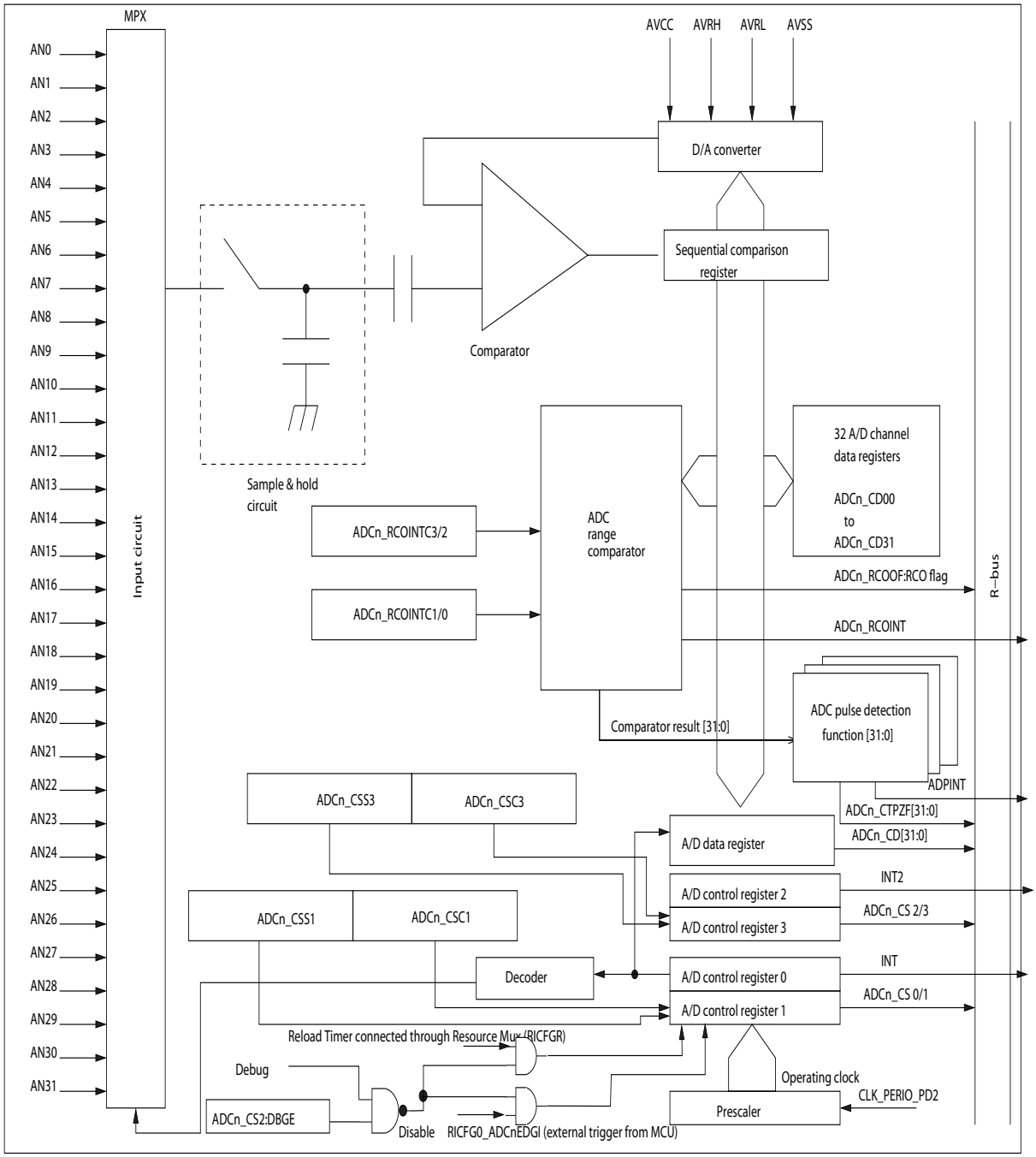
The A/D Converter converts analog input voltages into digital values. The A/D Converter features range comparator, 32 separate result data registers, four range comparators, and 32 pulse detection units.

Features of the A/D Converter

- Conversion time: minimum 1 μs per channel
- RC type successive approximation conversion with sample & hold circuit
- 10-bit or 8-bit resolution
- Programmable analog input selection from 32 channels
- One common result data register and 32 dedicated channel result data registers
- Scan conversion modes, continuous conversion of multiple channels, programmable for up to 32 channels:
  - Continuous mode, repeatedly converts the specified channels
  - Single mode, converts the specified channel(s) only once
- Stop mode, conversion of one channel, then temporarily halts until the next activation. (enables synchronization of the conversion start timing)
- Interrupt request generation to CPU for:
  - End of conversion interrupt (single channel)
  - End of scan interrupt (all enabled channels), results are stored in dedicated result registers
- A/D conversion can generate a DMA transfer request to transfer the results of A/D conversion to memory
- Conversion startup is possible by software, external trigger (falling edge), or timer
- Four range comparator channels, comparing the upper 8-bit of the conversion result
- Programmable upper and lower thresholds, individually for each comparator
- Any of the available ADC channel can be assigned to one of the four range comparators
- The comparison results will set flags per ADC channel, depending on the configuration. Possible configurations:
  - 'Outside range': The flags are set if the A/D result is below the lower OR above the upper threshold
  - 'Inside range': The flags are set if the A/D result is above the lower AND below the upper threshold
- The configuration can be set individually per ADC channel
- Range comparison triggers an A/D range comparator interrupt request to CPU
- Results of the range comparator can be filtered to ignore multiple spikes
- Debug mode is supported

Block diagram of A/D Converter

Figure 36-1. Block diagram of A/D Converter



ADC range comparator

The ADC range comparator has four comparison groups with an upper and a lower threshold register each. The 32 ADC channels can be enabled for range comparison and assigned to one of the four comparators individually.

If ADC range comparator is enabled for an ADC channel, it provides two result flags:

- Interrupt flag ADCn_RCOINT32, ADCn_RCOINT10:RCOINT, signalling that the ADC result is outside the range or inside the range ('inverted' operation)

- Over threshold flag ADCn_RCOOF32, ADCn_RCOOF10:RCOOF, showing that the value exceeded the upper threshold

Furthermore, each ADC channel can be enabled to send an interrupt request to the CPU, if the ADCn_RCOINT32, ADCn_RCOINT10:RCOINT flag is set.

ADC pulse detection function

The result of the comparison from the range comparator can be filtered by the ADC pulse detection function.

The ADC pulse detection function has a pair of reload registers to store the initial value for the positive and negative down counter. It has the status registers and interrupt registers corresponding to each analog input channel of the ADC.

The positive and the negative counters decrement on positive and negative events obtained from the result of the comparison from the range comparator.

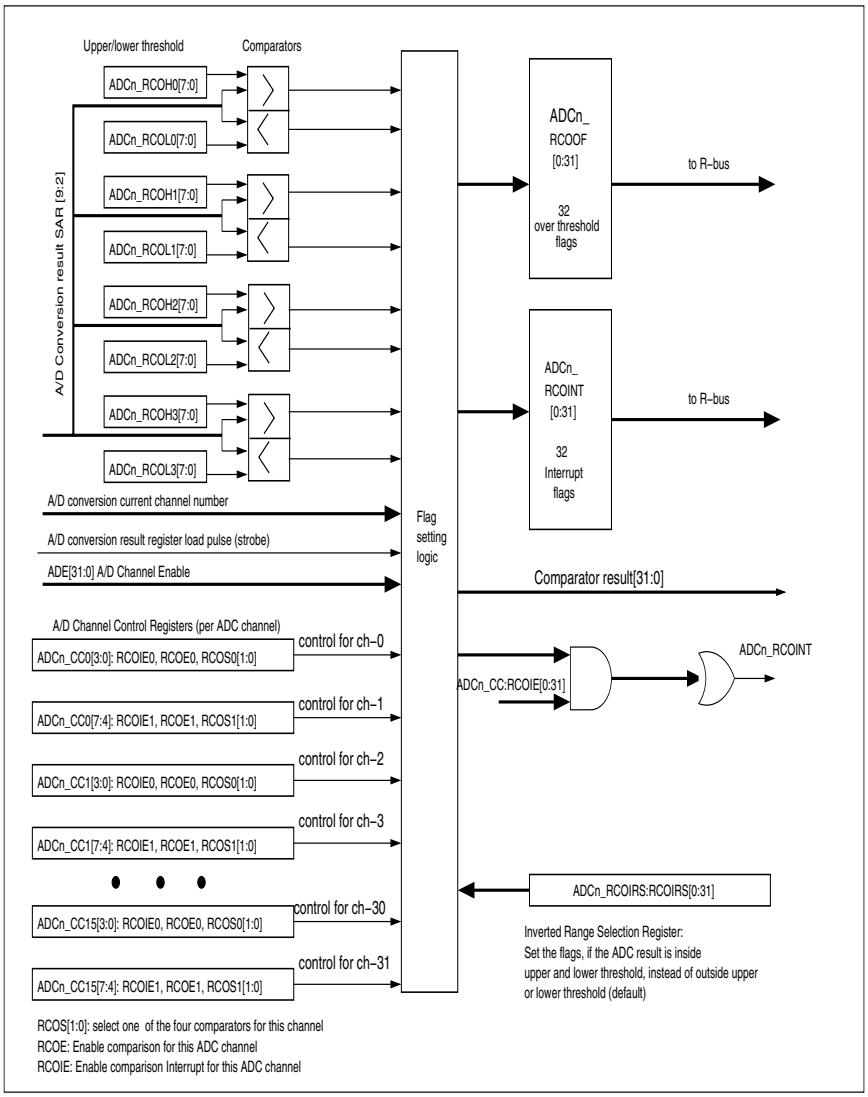
The features of ADC pulse detection function are:

- Detect events with desired length
- Filter parasitic inverted events
- Each ADC channel has a pulse detection function module associated with it
- Interrupt can be generated on detection of a pulse

For further details on

ADC pulse detection function and positive event/negative event refer to ADC pulse detection function.

Figure 36-2. Range comparator structure



36.2 A/D Converter registers

The A/D Converter contains registers to configure the operation of A/D conversion and to store the converted values. It also contains registers to configure the range comparators and registers to control and store the status of ADC pulse detection function. This section describes the registers of the A/D Converter in detail.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of A/D Converter

The following registers are available for each instance of A/D Converter:

- A/D Input Enable Registers (ADCn_ER32, ADCn_ER10)
- A/D Control Status Register 0 (ADCn_CS0)
- A/D Control Status Register 1 (ADCn_CS1)
- A/D Control Status Register 2 (ADCn_CS2)
- A/D Control Status Register 3 (ADCn_CS3)
- A/D Control Status Register 1 Set Register (ADCn_CSS1)
- A/D Control Status Register 1 Clear Register (ADCn_CSC1)
- A/D Control Status Register 3 Set Register (ADCn_CSS3)
- A/D Control Status Register 3 Clear Register (ADCn_CSC3)
- Common Data Register (ADCn_CR)
- Dedicated A/D Channel Data Register (ADCn_CD31~0)
- ADC Conversion Time Setting Register (ADCn_CT)
- A/D Start Channel Setting Register (ADCn_SCH)
- A/D End Channel Setting Register (ADCn_ECH)
- A/D Converter DMA Configuration Register (ADCn_MAR)
- A/D Converter DMA Configuration Set Register (ADCn_MASR)
- A/D Converter DMA Configuration Clear Register (ADCn_MACR)
- Range Comparator Upper Threshold Registers (ADCn_RCOH3~0)
- Range Comparator Lower Threshold Registers (ADCn_RCOL3~0)
- A/D Converter Channel Control Registers (ADCn_CC15~0)
- Inverted Range Selection Registers (ADCn_RCOIRS32, ADCn_RCOIRS10)
- Range Comparator Interrupt Flags (ADCn_RCOINT32, ADCn_RCOINT10)
- Range Comparator Over Threshold Flags (ADCn_RCOOF32, ADCn_RCOOF10)
- Range Comparator Interrupt Clear Registers (ADCn_RCOINTC32, ADCn_RCOINTC10)
- ADC Pulse Counter Reload Registers (ADCn_PCTNRL31~0/ADCn_PCTPRL31~0)
- ADC Pulse Counters (ADCn_PCTNCT31~0/ADCn_PCTPCT31~0)
- ADC Pulse Counter Zero Flag Registers (ADCn_PCZF32, ADCn_PCZF10)
- ADC Pulse Counter Zero Flag Clear Registers (ADCn_PCZFC32, ADCn_PCZFC10)
- ADC Pulse Counter Interrupt Enable Registers (ADCn_PCIE32, ADCn_PCIE10)
- ADC Pulse Counter Interrupt Enable Set Registers (ADCn_PCIES32, ADCn_PCIES10)
- ADC Pulse Counter Interrupt Enable Clear Registers (ADCn_PCIEC32, ADCn_PCIEC10)

Memory layout of A/D Converter registers

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x00000000	ADCn_ER32 00000000 00000000	
0x00000002	ADCn_ER10 00000000 00000000	
0x00000004	ADCn_CS1 00000000	ADCn_CS0 00000000
0x00000006	ADCn_CS3 00000000	ADCn_CS2 00000000
0x00000008	ADCn_CSS1 00000000	reserved XXXXXXXX
0x0000000A	ADCn_CSC1 00000000	reserved XXXXXXXX
0x0000000C	ADCn_CSS3 00000000	reserved XXXXXXXX
0x0000000E	ADCn_CSC3 00000000	reserved XXXXXXXX
0x00000010	ADCn_CR 000000XX XXXXXXXX	
0x00000012	reserved XXXXXXXX	reserved XXXXXXXX
0x00000014	reserved XXXXXXXX	reserved XXXXXXXX
0x00000016	reserved XXXXXXXX	reserved XXXXXXXX
0x00000018	ADCn_CD0 000000XX XXXXXXXX	
0x0000001A	ADCn_CD1 000000XX XXXXXXXX	
0x0000001C	ADCn_CD2 000000XX XXXXXXXX	
0x0000001E	ADCn_CD3 000000XX XXXXXXXX	
0x00000020	ADCn_CD4 000000XX XXXXXXXX	
0x00000022	ADCn_CD5 000000XX XXXXXXXX	

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x00000024	ADCn_CD6 000000XX XXXXXXXX	
0x00000026	ADCn_CD7 000000XX XXXXXXXX	
0x00000028	ADCn_CD8 000000XX XXXXXXXX	
0x0000002A	ADCn_CD9 000000XX XXXXXXXX	
0x0000002C	ADCn_CD10 000000XX XXXXXXXX	
0x0000002E	ADCn_CD11 000000XX XXXXXXXX	
0x00000030	ADCn_CD12 000000XX XXXXXXXX	
0x00000032	ADCn_CD13 000000XX XXXXXXXX	
0x00000034	ADCn_CD14 000000XX XXXXXXXX	
0x00000036	ADCn_CD15 000000XX XXXXXXXX	
0x00000038	ADCn_CD16 000000XX XXXXXXXX	
0x0000003A	ADCn_CD17 000000XX XXXXXXXX	
0x0000003C	ADCn_CD18 000000XX XXXXXXXX	
0x0000003E	ADCn_CD19 000000XX XXXXXXXX	
0x00000040	ADCn_CD20 000000XX XXXXXXXX	
0x00000042	ADCn_CD21 000000XX XXXXXXXX	
0x00000044	ADCn_CD22 000000XX XXXXXXXX	
0x00000046	ADCn_CD23 000000XX XXXXXXXX	
0x00000048	ADCn_CD24 000000XX XXXXXXXX	

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x0000004A	ADCn_CD25 000000XX XXXXXXXX	
0x0000004C	ADCn_CD26 000000XX XXXXXXXX	
0x0000004E	ADCn_CD27 000000XX XXXXXXXX	
0x00000050	ADCn_CD28 000000XX XXXXXXXX	
0x00000052	ADCn_CD29 000000XX XXXXXXXX	
0x00000054	ADCn_CD30 000000XX XXXXXXXX	
0x00000056	ADCn_CD31 000000XX XXXXXXXX	
0x00000058	reserved XXXXXXXX	reserved XXXXXXXX
0x0000005A	reserved XXXXXXXX	reserved XXXXXXXX
0x0000005C	reserved XXXXXXXX	reserved XXXXXXXX
0x0000005E	ADCn_CT 00010000 00101100	
0x00000060	ADCn_ECH 00000000	ADCn_SCH 00000000
0x00000062	reserved XXXXXXXX	ADCn_MAR 00000000
0x00000064	reserved XXXXXXXX	ADCn_MACR 00000000
0x00000066	reserved XXXXXXXX	ADCn_MASR 00000000
0x00000068	ADCn_RCOH0 11111111	ADCn_RCOL0 00000000
0x0000006A	ADCn_RCOH1 11111111	ADCn_RCOL1 00000000
0x0000006C	ADCn_RCOH2 11111111	ADCn_RCOL2 00000000
0x0000006E	ADCn_RCOH3 11111111	ADCn_RCOL3 00000000

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x00000070	ADCn_CC1 00000000	ADCn_CC0 00000000
0x00000072	ADCn_CC3 00000000	ADCn_CC2 00000000
0x00000074	ADCn_CC5 00000000	ADCn_CC4 00000000
0x00000076	ADCn_CC7 00000000	ADCn_CC6 00000000
0x00000078	ADCn_CC9 00000000	ADCn_CC8 00000000
0x0000007A	ADCn_CC11 00000000	ADCn_CC10 00000000
0x0000007C	ADCn_CC13 00000000	ADCn_CC12 00000000
0x0000007E	ADCn_CC15 00000000	ADCn_CC14 00000000
0x00000080	ADCn_RCOIRS32 00000000 00000000	
0x00000082	ADCn_RCOIRS10 00000000 00000000	
0x00000084	ADCn_RCOOF32 00000000 00000000	
0x00000086	ADCn_RCOOF10 00000000 00000000	
0x00000088	ADCn_RCOINT32 00000000 00000000	
0x0000008A	ADCn_RCOINT10 00000000 00000000	
0x0000008C	ADCn_RCOINTC32 00000000 00000000	
0x0000008E	ADCn_RCOINTC10 00000000 00000000	
0x00000090	ADCn_PCTNRL0 00000000	ADCn_PCTPRL0 00000000
0x00000092	ADCn_PCTNCT0 00000000	ADCn_PCTPCT0 00000000
0x00000094	ADCn_PCTNRL1 00000000	ADCn_PCTPRL1 00000000

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x00000096	ADCn_PCTNCT1 00000000	ADCn_PCTPCT1 00000000
0x00000098	ADCn_PCTNRL2 00000000	ADCn_PCTPRL2 00000000
0x0000009A	ADCn_PCTNCT2 00000000	ADCn_PCTPCT2 00000000
0x0000009C	ADCn_PCTNRL3 00000000	ADCn_PCTPRL3 00000000
0x0000009E	ADCn_PCTNCT3 00000000	ADCn_PCTPCT3 00000000
0x000000A0	ADCn_PCTNRL4 00000000	ADCn_PCTPRL4 00000000
0x000000A2	ADCn_PCTNCT4 00000000	ADCn_PCTPCT4 00000000
0x000000A4	ADCn_PCTNRL5 00000000	ADCn_PCTPRL5 00000000
0x000000A6	ADCn_PCTNCT5 00000000	ADCn_PCTPCT5 00000000
0x000000A8	ADCn_PCTNRL6 00000000	ADCn_PCTPRL6 00000000
0x000000AA	ADCn_PCTNCT6 00000000	ADCn_PCTPCT6 00000000
0x000000AC	ADCn_PCTNRL7 00000000	ADCn_PCTPRL7 00000000
0x000000AE	ADCn_PCTNCT7 00000000	ADCn_PCTPCT7 00000000
0x000000B0	ADCn_PCTNRL8 00000000	ADCn_PCTPRL8 00000000
0x000000B2	ADCn_PCTNCT8 00000000	ADCn_PCTPCT8 00000000
0x000000B4	ADCn_PCTNRL9 00000000	ADCn_PCTPRL9 00000000
0x000000B6	ADCn_PCTNCT9 00000000	ADCn_PCTPCT9 00000000
0x000000B8	ADCn_PCTNRL10 00000000	ADCn_PCTPRL10 00000000
0x000000BA	ADCn_PCTNCT10 00000000	ADCn_PCTPCT10 00000000

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x000000BC	ADCn_PCTNRL11 00000000	ADCn_PCTPRL11 00000000
0x000000BE	ADCn_PCTNCT11 00000000	ADCn_PCTPCT11 00000000
0x000000C0	ADCn_PCTNRL12 00000000	ADCn_PCTPRL12 00000000
0x000000C2	ADCn_PCTNCT12 00000000	ADCn_PCTPCT12 00000000
0x000000C4	ADCn_PCTNRL13 00000000	ADCn_PCTPRL13 00000000
0x000000C6	ADCn_PCTNCT13 00000000	ADCn_PCTPCT13 00000000
0x000000C8	ADCn_PCTNRL14 00000000	ADCn_PCTPRL14 00000000
0x000000CA	ADCn_PCTNCT14 00000000	ADCn_PCTPCT14 00000000
0x000000CC	ADCn_PCTNRL15 00000000	ADCn_PCTPRL15 00000000
0x000000CE	ADCn_PCTNCT15 00000000	ADCn_PCTPCT15 00000000
0x000000D0	ADCn_PCTNRL16 00000000	ADCn_PCTPRL16 00000000
0x000000D2	ADCn_PCTNCT16 00000000	ADCn_PCTPCT16 00000000
0x000000D4	ADCn_PCTNRL17 00000000	ADCn_PCTPRL17 00000000
0x000000D6	ADCn_PCTNCT17 00000000	ADCn_PCTPCT17 00000000
0x000000D8	ADCn_PCTNRL18 00000000	ADCn_PCTPRL18 00000000
0x000000DA	ADCn_PCTNCT18 00000000	ADCn_PCTPCT18 00000000
0x000000DC	ADCn_PCTNRL19 00000000	ADCn_PCTPRL19 00000000
0x000000DE	ADCn_PCTNCT19 00000000	ADCn_PCTPCT19 00000000
0x000000E0	ADCn_PCTNRL20 00000000	ADCn_PCTPRL20 00000000

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x000000E2	ADCn_PCTNCT20 00000000	ADCn_PCTPCT20 00000000
0x000000E4	ADCn_PCTNRL21 00000000	ADCn_PCTPRL21 00000000
0x000000E6	ADCn_PCTNCT21 00000000	ADCn_PCTPCT21 00000000
0x000000E8	ADCn_PCTNRL22 00000000	ADCn_PCTPRL22 00000000
0x000000EA	ADCn_PCTNCT22 00000000	ADCn_PCTPCT22 00000000
0x000000EC	ADCn_PCTNRL23 00000000	ADCn_PCTPRL23 00000000
0x000000EE	ADCn_PCTNCT23 00000000	ADCn_PCTPCT23 00000000
0x000000F0	ADCn_PCTNRL24 00000000	ADCn_PCTPRL24 00000000
0x000000F2	ADCn_PCTNCT24 00000000	ADCn_PCTPCT24 00000000
0x000000F4	ADCn_PCTNRL25 00000000	ADCn_PCTPRL25 00000000
0x000000F6	ADCn_PCTNCT25 00000000	ADCn_PCTPCT25 00000000
0x000000F8	ADCn_PCTNRL26 00000000	ADCn_PCTPRL26 00000000
0x000000FA	ADCn_PCTNCT26 00000000	ADCn_PCTPCT26 00000000
0x000000FC	ADCn_PCTNRL27 00000000	ADCn_PCTPRL27 00000000
0x000000FE	ADCn_PCTNCT27 00000000	ADCn_PCTPCT27 00000000
0x00000100	ADCn_PCTNRL28 00000000	ADCn_PCTPRL28 00000000
0x00000102	ADCn_PCTNCT28 00000000	ADCn_PCTPCT28 00000000
0x00000104	ADCn_PCTNRL29 00000000	ADCn_PCTPRL29 00000000
0x00000106	ADCn_PCTNCT29 00000000	ADCn_PCTPCT29 00000000

Table 36-1. Memory layout of A/D Converter registers

Offset	+1	+0
0x00000108	ADCn_PCTNRL30 00000000	ADCn_PCTPRL30 00000000
0x0000010A	ADCn_PCTNCT30 00000000	ADCn_PCTPCT30 00000000
0x0000010C	ADCn_PCTNRL31 00000000	ADCn_PCTPRL31 00000000
0x0000010E	ADCn_PCTNCT31 00000000	ADCn_PCTPCT31 00000000
0x00000110	ADCn_PCZF10 00000000 00000000	
0x00000112	ADCn_PCZF32 00000000 00000000	
0x00000114	ADCn_PCZFC10 00000000 00000000	
0x00000116	ADCn_PCZFC32 00000000 00000000	
0x00000118	ADCn_PCIE10 00000000 00000000	
0x0000011A	ADCn_PCIE32 00000000 00000000	
0x0000011C	ADCn_PCIES10 00000000 00000000	
0x0000011E	ADCn_PCIES32 00000000 00000000	
0x00000120	ADCn_PCIEC10 00000000 00000000	
0x00000122	ADCn_PCIEC32 00000000 00000000	
0x00000124	reserved XXXXXXXX XXXXXXXX	

36.2.1 A/D Input Enable Registers (ADCn_ER32, ADCn_ER10)

These registers enable the analog input functions of the A/D Converter. There are two 16-bit registers for enabling 32 analog channels ADCn_ER32, ADCn_ER10.

A/D Input Enable Registers (ADCn_ER32)

Figure 36-3. A/D Input Enable Registers (ADCn_ER32)

ADCn_ER32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ADE0	ADE1	ADE2	ADE3	ADE4	ADE5	ADE6	ADE7	ADE8	ADE9	ADE10	ADE11	ADE12	ADE13	ADE14	ADE15
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-2. A/D Input Enable Registers (ADCn_ER32) bits

Bit Position	Bit Field Name	Bit Description
[15]	ADE0	A/D Input Enable for ADC Channel 0 '0': ADC channel 0 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 0 '1': ADC channel 0 is enabled The digital input buffer of the corresponding pin is disabled
[14]	ADE1	A/D Input Enable for ADC Channel 1 '0': ADC channel 1 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 1 '1': ADC channel 1 is enabled The digital input buffer of the corresponding pin is disabled
[13]	ADE2	A/D Input Enable for ADC Channel 2 '0': ADC channel 2 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 2 '1': ADC channel 2 is enabled The digital input buffer of the corresponding pin is disabled

Table 36-2. A/D Input Enable Registers (ADCn_ER32) bits

Bit Position	Bit Field Name	Bit Description
[12]	ADE3	A/D Input Enable for ADC Channel 3 '0': ADC channel 3 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 3 '1': ADC channel 3 is enabled The digital input buffer of the corresponding pin is disabled
[11]	ADE4	A/D Input Enable for ADC Channel 4 '0': ADC channel 4 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 4 '1': ADC channel 4 is enabled The digital input buffer of the corresponding pin is disabled
[10]	ADE5	A/D Input Enable for ADC Channel 5 '0': ADC channel 5 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 5 '1': ADC channel 5 is enabled The digital input buffer of the corresponding pin is disabled
[9]	ADE6	A/D Input Enable for ADC Channel 6 '0': ADC channel 6 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 6 '1': ADC channel 6 is enabled The digital input buffer of the corresponding pin is disabled
[8]	ADE7	A/D Input Enable for ADC Channel 7 '0': ADC channel 7 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 7 '1': ADC channel 7 is enabled The digital input buffer of the corresponding pin is disabled
[7]	ADE8	A/D Input Enable for ADC Channel 8 '0': ADC channel 8 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 8 '1': ADC channel 8 is enabled The digital input buffer of the corresponding pin is disabled
[6]	ADE9	A/D Input Enable for ADC Channel 9 '0': ADC channel 6 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 9 '1': ADC channel 9 is enabled The digital input buffer of the corresponding pin is disabled

Table 36-2. A/D Input Enable Registers (ADCn_ER32) bits

Bit Position	Bit Field Name	Bit Description
[5]	ADE10	A/D Input Enable for ADC Channel 10 '0': ADC channel 10 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 10 '1': ADC channel 10 is enabled The digital input buffer of the corresponding pin is disabled
[4]	ADE11	A/D Input Enable for ADC Channel 11 '0': ADC channel 11 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 11 '1': ADC channel 11 is enabled The digital input buffer of the corresponding pin is disabled
[3]	ADE12	A/D Input Enable for ADC Channel 12 '0': ADC channel 12 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 12 '1': ADC channel 12 is enabled The digital input buffer of the corresponding pin is disabled
[2]	ADE13	A/D Input Enable for ADC Channel 13 '0': ADC channel 13 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 13 '1': ADC channel 13 is enabled The digital input buffer of the corresponding pin is disabled
[1]	ADE14	A/D Input Enable for ADC Channel 14 '0': ADC channel 14 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 14 '1': ADC channel 14 is enabled The digital input buffer of the corresponding pin is disabled
[0]	ADE15	A/D Input Enable for ADC Channel 15 '0': ADC channel 15 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 15 '1': ADC channel 15 is enabled The digital input buffer of the corresponding pin is disabled

A/D Input Enable Registers (ADCn_ER10)

Figure 36-4. A/D Input Enable Registers (ADCn_ER10)

ADCn_ER10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ADE16	ADE17	ADE18	ADE19	ADE20	ADE21	ADE22	ADE23	ADE24	ADE25	ADE26	ADE27	ADE28	ADE29	ADE30	ADE31
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-3. A/D Input Enable Registers (ADCn_ER10) bits

Bit Position	Bit Field Name	Bit Description
[15]	ADE16	A/D Input Enable ADC Channel 16 '0': ADC channel 16 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 16 '1': ADC channel 16 is enabled The digital input buffer of the corresponding pin is disabled
[14]	ADE17	A/D Input Enable ADC Channel 17 '0': ADC channel 17 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 17 '1': ADC channel 17 is enabled The digital input buffer of the corresponding pin is disabled
[13]	ADE18	A/D Input Enable ADC Channel 18 '0': ADC channel 18 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 18 '1': ADC channel 18 is enabled The digital input buffer of the corresponding pin is disabled
[12]	ADE19	A/D Input Enable ADC Channel 19 '0': ADC channel 19 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 19 '1': ADC channel 19 is enabled The digital input buffer of the corresponding pin is disabled

Table 36-3. A/D Input Enable Registers (ADCn_ER10) bits

Bit Position	Bit Field Name	Bit Description
[11]	ADE20	A/D Input Enable ADC Channel 20 '0': ADC channel 20 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 20 '1': ADC channel 20 is enabled The digital input buffer of the corresponding pin is disabled
[10]	ADE21	A/D Input Enable ADC Channel 21 '0': ADC channel 21 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 21 '1': ADC channel 21 is enabled The digital input buffer of the corresponding pin is disabled
[9]	ADE22	A/D Input Enable ADC Channel 22 '0': ADC channel 22 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 22 '1': ADC channel 22 is enabled The digital input buffer of the corresponding pin is disabled
[8]	ADE23	A/D Input Enable ADC Channel 23 '0': ADC channel 23 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 23 '1': ADC channel 23 is enabled The digital input buffer of the corresponding pin is disabled
[7]	ADE24	A/D Input Enable ADC Channel 24 '0': ADC channel 24 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 24 '1': ADC channel 24 is enabled The digital input buffer of the corresponding pin is disabled
[6]	ADE25	A/D Input Enable ADC Channel 25 '0': ADC channel 25 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 25 '1': ADC channel 25 is enabled The digital input buffer of the corresponding pin is disabled

Table 36-3. A/D Input Enable Registers (ADCn_ER10) bits

Bit Position	Bit Field Name	Bit Description
[5]	ADE26	A/D Input Enable ADC Channel 26 '0': ADC channel 26 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 26 '1': ADC channel 26 is enabled The digital input buffer of the corresponding pin is disabled
[4]	ADE27	A/D Input Enable ADC Channel 27 '0': ADC channel 27 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 27 '1': ADC channel 27 is enabled The digital input buffer of the corresponding pin is disabled
[3]	ADE28	A/D Input Enable ADC Channel 28 '0': ADC channel 28 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 28 '1': ADC channel 28 is enabled The digital input buffer of the corresponding pin is disabled
[2]	ADE29	A/D Input Enable ADC Channel 29 '0': ADC channel 29 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 29 '1': ADC channel 29 is enabled The digital input buffer of the corresponding pin is disabled
[1]	ADE30	A/D Input Enable ADC Channel 30 '0': ADC channel 30 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 30 '1': ADC channel 30 is enabled The digital input buffer of the corresponding pin is disabled
[0]	ADE31	A/D Input Enable for ADC Channel 31 '0': ADC channel 31 is disabled It is skipped in the ADC scan. Do not set either ADCn_SCH:ANS or ADCn_ECH:ANE to channel 31 '1': ADC channel 31 is enabled The digital input buffer of the corresponding pin is disabled

Notes:

1. Ensure to set start channel and end channel to cover all enabled channels.
2. All the bits in the registers ADCn_ER32 and ADCn_ER10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.2.2 A/D Control Status Register 0 (ADCn_CS0)

The A/D Control Status Register 0 configures the A/D Converter mode and the resolution of the ADC. It also shows the converted analog channel.

A/D Control Status Register 0 (ADCn_CS0)

Figure 36-5. A/D Control Status Register 0 (ADCn_CS0)

ADCn_CS0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
MD[1]	MD[0]	S10	ACH[4]	ACH[3]	ACH[2]	ACH[1]	ACH[0]
RpWp	RpWp	RpWp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 36-4. A/D Control Status Register 0 (ADCn_CS0) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	MD	A/D Converter Mode Set '00': Single mode 1 (reactivation during A/D conversion is allowed) '01': Single mode 2 (reactivation during A/D conversion is not allowed) '10': Continuous mode (reactivation during A/D conversion is not allowed) '11': Stop mode (reactivation during A/D conversion is not allowed) Single mode: A/D conversion is sequentially performed from the selected start channel (ADCn_SCH) to the selected end channel (ADCn_ECH). The conversion stops once it has been done for all enabled channels within this range. Continuous mode: A/D conversion is repeatedly performed from the selected start channel to the selected end channel continuously. Stop mode: A/D conversion is performed from the selected start channel to the selected end channel, followed by a pause after each channel. The conversion is resumed by a new trigger event. When A/D conversion is started in continuous mode or stop mode, conversion operation can be stopped by clearing ADCn_CS1:BUSY bit by writing ADCn_CSC1:BUSYC to '1'. After forcibly stopping and re-starting (writing ADCn_CSC1:STRT to '1'), conversion starts from the start channel, selected by ADCn_SCH. Reactivation during A/D conversion is disabled for any of the timer, external trigger, and software start sources in single mode 2, continuous, and stop mode.
[5]	S10	Resolution of A/D Conversion '0': 10-bit resolution '1': 8-bit resolution Conversion result is stored in lower 8 bits of ADCn_CR and ADCn_CD31~0.
[4:0]	ACH	Analog Convert Select Channel These bits show the channel number of the currently or previously converted analog channel, depending on ADCn_CS1:ACHMD. '00000':Channel converted is AN0 '00001':Channel converted is AN1 '00010':Channel converted is AN2 .... '11111':Channel converted is AN31

36.2.3 A/D Control Status Register 1 (ADCn_CS1)

The A/D Control Status Register 1 shows the status of the end of conversion interrupt along with the status of ADCn_CS1:BUSY and ADCn_CS1:PAUS bits. This register can configure the trigger source for the ADC, software trigger for ADC, and the mode for the ADCn_CS0:ACH bits.

A/D Control Status Register 1 (ADCn_CS1)

Figure 36-6. A/D Control Status Register 1 (ADCn_CS1)

ADCn_CS1							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BUSY	INT	INTE	PAUS	STS[1]	STS[0]	STRT	ACHMD
Rp	Rp	RpWp	Rp	RpWp	RpWp	Rp0Wp1	RpWp
0	0	0	0	0	0	0	0

Table 36-5. A/D Control Status Register 1 (ADCn_CS1) bits

Bit Position	Bit Field Name	Bit Description
[7]	BUSY	Busy Flag and Stop This bit indicates the A/D Converter operation. It is set on activation of A/D conversion and cleared on completion of conversion in single and stop conversion mode. This bit is cleared by writing '1' in ADCn_CSC1:BUSYC bit. If this bit is cleared during A/D conversion, it terminates the conversion. This is how conversion is forcibly terminated in continuous and stop modes.
[6]	INT	End of Conversion Interrupt Flag This bit is set when conversion data is stored in ADCn_CR. If ADCn_CS1:INTE is set an interrupt request is generated. This bit is cleared by writing '1' in ADCn_CSC1:INTC. When ADCn_MAR:DRQEN = '1' (DMA is enabled) and ADCn_CS2:DPDIS = '0' (data protection enabled) this flag is cleared after the result register ADCn_CR has been read (except by the debug master, DAP).

Table 36-5. A/D Control Status Register 1 (ADCn_CS1) bits

Bit Position	Bit Field Name	Bit Description
[5]	INTE	End of Conversion Interrupt Enable '0': Interrupt is disabled '1': Interrupt is enabled This bit can also be set by ADCn_CSS1:INTES or cleared by ADCn_CSC1:INTEC.
[4]	PAUS	A/D Converter Pause The PAUS bit signifies that the ADC conversion was paused since the last time when this bit was cleared. When a conversion has finished and if the data protection function is enabled (ADCn_CS2:DPDIS = '0'), the ADCn_CS1:PAUS bit will be set if any of the below-mentioned conditions are true: *When 'end of scan' interrupt is active (ADCn_CS3:INTE2 = '1' and ADCn_CS3:INT2 = '1') and the conversion data of the start channel (set by ADCn_SCH) is ready for writing to the registers. *When 'end of conversion' interrupt is active (ADCn_CS1:INTE = '1' and INT = '1') and the conversion data of any channel is ready for writing to the registers. *When DMA operation for 'end of conversion' is enabled (ADCn_MAR:DRQEN = '1') and 'end of conversion' interrupt flag is active (INT = '1') and the conversion data of any channel is ready for writing to the registers. As long as any of the above conditions are true, the current conversion result is not stored in the Common Data Register ADCn_CR and the applicable Channel Data Register ADCn_CD31~0, and the ADC conversion is paused. As soon as the above conditions disappear, the current conversion result is stored in the registers and the conversion continues, but the ADCn_CS1:PAUS bit remains set. The ADCn_CS1:PAUS bit is cleared by writing '1' to ADCn_CSC1:PAUSC bit.

Table 36-5. A/D Control Status Register 1 (ADCn_CS1) bits

Bit Position	Bit Field Name	Bit Description
[3:2]	STS	Start Source Select These bits select the A/D activation source. '00': Software activation (initial value), '01': External trigger (selected by RICFG0_ADCnEDGI register) activation and software activation, '10': Timer (selected by RICFG0_ADCnTIMI) activation and software activation, '11': External trigger (selected by RICFG0_ADCnEDGI register) activation, timer (selected by RICFG0_ADCnTIMI) activation and software activation, In multiple-activation modes, the first activation to occur starts A/D conversion. The activation source changes immediately on writing to the register. Therefore care is required when switching activation mode during A/D operation. The A/D Converter detects falling edges on the external trigger pin. When external trigger level is 'L' and if these bits are changed to external trigger activation mode, A/D converting may start.
[1]	STRT	Start Writing '1' to this bit starts A/D conversion. Write '1' again to restart conversion. In continuous and stop mode, restarting does not occur. Check ADCn_CS1:BUSY bit before writing '1' (activate conversion after clearing). Reading this bit always returns '0'.
[0]	ACHMD	ACH Register Mode For reading out ADCn_CS0:ACH[4:0] register bits, there is a direct mode and a latched mode. In direct mode, ADCn_CS0:ACH[4:0] shows the ADC channel which is currently in conversion, i.e. the internal conversion channel pointer. This pointer is incremented immediately after a conversion is finished. In latched mode, ADCn_CS0:ACH[4:0] shows the ADC channel whose conversion was finished previously. After a conversion is finished, the conversion channel pointer is latched and the latched data can be read in this mode. At the end of the next conversion, the latch is overwritten if no PAUSE condition exists. '0': Direct ACH register mode '1': Latched ACH register mode

36.2.4 A/D Control Status Register 2 (ADCn_CS2)

The A/D Control Status Register 2 has bits to enable the data protection function and the debug functionality.

A/D Control Status Register 2 (ADCn_CS2)

Figure 36-7. A/D Control Status Register 2 (ADCn_CS2)

ADCn_CS2							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	DPDIS	DBGE
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-6. A/D Control Status Register 2 (ADCn_CS2) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	DPDIS	Data Protection Disable '0': Data protection function is enabled '1': Data protection function is disabled
[0]	DBGE	Debug Enable This bit enables the debug functionality if set. '0': Debug functionality is disabled '1': Debug functionality is enabled When ADC enters into debug mode the conversion is stopped after the conversion is finished and ADC holds its state" For more details refer to 36.3 Operation of A/D Converter.

36.2.5 A/D Control Status Register 3 (ADCn_CS3)

The A/D Control Status Register 3 shows the status of the end of scan and end of conversion interrupt along with the status of ADCn_CS1:BUSY and ADCn_CS1:PAUS bits. It controls the enabling of the end of scan interrupt.

A/D Control Status Register 3 (ADCn_CS3)

Figure 36-8. A/D Control Status Register 3 (ADCn_CS3)

ADCn_CS3							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BUSY	INT	INTE	PAUS	read0	read0	INT2	INTE2
Rp	Rp	Rp	Rp	Rp0	Rp0	Rp	RpWp
0	0	0	0	0	0	0	0

Table 36-7. A/D Control Status Register 3 (ADCn_CS3) bits

Bit Position	Bit Field Name	Bit Description
[7]	BUSY	Mirror of BUSY bit in ADCn_CS1 This bit is a mirror of the corresponding bit in ADCn_CS1:BUSY. Intended to read out the status of the corresponding bit in the ADCn_CS1 register.
[6]	INT	Mirror of INT bit in ADCn_CS1 This bit is a mirror of the corresponding bit in ADCn_CS1:INT. Intended to read out the interrupt information in the ADCn_CS1 register.
[5]	INTE	Mirror of INTE bit in ADCn_CS1 This bit is a mirror of the corresponding bit in ADCn_CS1:INTE. Intended to read out the status of the corresponding bit from ADCn_CS1 register.
[4]	PAUS	Mirror of PAUS bit in ADCn_CS1 This bit is a mirror of the corresponding bit in ADCn_CS1:PAUS. Intended to read out the status of the corresponding bit from ADCn_CS1 register.
[3:2]	read0	-

Table 36-7. A/D Control Status Register 3 (ADCn_CS3) bits

Bit Position	Bit Field Name	Bit Description
[1]	INT2	End of Scan Interrupt Flag This bit is set when conversion data in continuous mode is stored in ADCn_CR, where the last channel is defined by ADCn_ECH register. If ADCn_CS3:INTE2 is set an end of scan interrupt request is generated. This bit is cleared when ADCn_CSC3:INT2C = '1'.
[0]	INTE2	Enable End of Scan Interrupt This bit enables the end of scan interrupt (INT2). '0': End of scan interrupt is disabled '1': End of scan interrupt is enabled

36.2.6 A/D Control Status Register 1 Set Register (ADCn_CSS1)

The A/D Control Status Register 1 Set Register (ADCn_CSS1) contains bits to set the bits A/D Control Status Register 1 (ADCn_CS1).

A/D Control Status Register 1 Set Register (ADCn_CSS1)

Figure 36-9. A/D Control Status Register 1 Set Register (ADCn_CSS1)

ADCn_CSS1							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	INTES	read0	read0	read0	STRTS	read0
Rp0	Rp0	Rp0Wp1	Rp0	Rp0	Rp0	Rp0Wp1	Rp0
0	0	0	0	0	0	0	0

Table 36-8. A/D Control Status Register 1 Set Register (ADCn_CSS1) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5]	INTES	End of Conversion Interrupt Enable Set '0': No effect '1': Sets ADCn_CS1:INTE bit Reading this bit always returns '0'.
[4:2]	read0	-
[1]	STRTS	Start Set '0': No effect '1': Sets ADCn_CS1:STRT bit Reading this bit always returns '0'.
[0]	read0	-

36.2.7 A/D Control Status Register 1 Clear Register (ADCn_CSC1)

The A/D Control Status Register 1 Clear Register contains bits to clear the bits A/D Control Status Register 1 (ADCn_CS1).

A/D Control Status Register 1 Clear Register (ADCn_CSC1)

Figure 36-10. A/D Control Status Register 1 Clear Register (ADCn_CSC1)

ADCn_CSC1							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BUSYC	INTC	INTEC	PAUSC	read0	read0	read0	read0
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0

Table 36-9. A/D Control Status Register 1 Clear Register (ADCn_CSC1) bits

Bit Position	Bit Field Name	Bit Description
[7]	BUSYC	BUSY Clear bit '0': No effect '1': Clears ADCn_CS1:BUSY bit Reading this bit always returns '0'.
[6]	INTC	End of Conversion Interrupt Clear '0': No effect '1': Clears ADCn_CS1:INT bit Reading this bit always returns '0'.
[5]	INTEC	End of Conversion Interrupt Enable Clear '0': No effect '1': Clears ADCn_CS1:INTE bit Reading this bit always returns '0'.
[4]	PAUSC	PAUS Clear bit '0': No effect '1': Clears ADCn_CS1:PAUS bit Reading this bit always returns '0'.
[3:0]	read0	-

36.2.8 A/D Control Status Register 3 Set Register (ADCn_CSS3)

The A/D Control Status Register 3 Set Register contains bits to set the bits of A/D Control Status Register 3 (ADCn_CS3).

A/D Control Status Register 3 Set Register (ADCn_CSS3)

Figure 36-11. A/D Control Status Register 3 Set Register (ADCn_CSS3)

ADCn_CSS3							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	INTE2S
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1
0	0	0	0	0	0	0	0

Table 36-10. A/D Control Status Register 3 Set Register (ADCn_CSS3) bits

Bit Position	Bit Field Name	Bit Description
[7:1]	read0	-
[0]	INTE2S	Enable End of Scan Interrupt Set bit '0': No effect '1': Sets ADCn_CS3:INTE2 bit Reading this bit always returns '0'.

36.2.9 A/D Control Status Register 3 Clear Register (ADCn_CSC3)

The A/D Control Status Register 3 Clear Register contains bits to clear the bits of A/D Control Status Register 3 (ADCn_CS3).

A/D Control Status Register 3 Clear Register (ADCn_CSC3)

Figure 36-12. A/D Control Status Register 3 Clear Register (ADCn_CSC3)

ADCn_CSC3							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	INT2C	INTE2C
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 36-11. A/D Control Status Register 3 Clear Register (ADCn_CSC3)

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	INT2C	End of Scan Interrupt Clear bit '0': No effect '1': Clears ADCn_CS3:INT2 bit Reading this bit always returns '0'.
[0]	INTE2C	Enable End of Scan Interrupt Clear bit '0': No effect '1': Clears ADCn_CS3:INTE2 bit Reading this bit always returns '0'.

36.2.10 Common Data Register (ADCn_CR)

This register stores the conversion results of the A/D Converter. The register values are updated at the completion of each conversion.

Common Data Register (ADCn_CR)

Figure 36-13. Common Data Register (ADCn_CR)

ADCn_CR															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X

Table 36-12. Common Data Register (ADCn_CR) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	read0	-
[9:0]	D	Common Data These bits store the conversion results of the A/D Converter. The register values are updated at the completion of each conversion. In 8-bit mode, the result is provided in bit[7:0]. In 10-bit mode, the result is provided in bit[9:0]. **Note:** When using 10-bit ADC resolution and DMA transfer, it is mandatory to configure the DMA for 16-bit access width to guarantee correct operation of the data protection function. A byte access during 10-bit mode may cause malfunction of data protection function during DMA access.

Note: The A/D Converter has a conversion data protection function. In continuous conversion mode, the protection function can be changed to protect the A/D Channel Data Registers rather than the A/D Data Register (ADCn_CR). For more information refer to 36.3 Operation of A/D Converter.

36.2.11 Dedicated A/D Channel Data Register (ADCn_CD31~0)

There are 32 ADC result data registers, one per channel. The registers are written by hardware at the end of conversion of the corresponding channel. ADCn_CD0 is attached to channel 0, ADCn_CD31 is attached to channel 31. Only ADCn_CD0 is described here. Other registers (ADCn_CD1, ADCn_CD2.... ADCn_CD31) have similar bit fields.

Dedicated A/D Channel Data Register 0 (ADCn_CD0)

Figure 36-14. Dedicated A/D Channel Data Register 0 (ADCn_CD0)

ADCn_CD0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X

Table 36-13. Dedicated A/D Channel Data Register 0 (ADCn_CD0) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	read0	-
[9:0]	D	Dedicated A/D Channel Data These bits store the conversion result of ADC channel 0. In 8-bit mode, the result is provided in bit[7:0]. In 10-bit mode, the result is provided in bit[9:0].

Note: The A/D Converter has a conversion data protection function. In continuous conversion mode, the protection function can be changed to protect the A/D Channel Data Registers rather than the A/D Data Register (ADCn_CR). For more information refer to 36.3 Operation of A/D Converter.

36.2.12 ADC Conversion Time Setting Register (ADCn_CT)

ADCn_CT register controls sampling time and comparison time of the A/D Converter. This register sets A/D conversion time. S/W should not update the value of this register during A/D conversion operation.

ADC Conversion Time Setting Register (ADCn_CT)

Figure 36-15. ADC Conversion Time Setting Register (ADCn_CT)

ADCn_CT															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CT[5]	CT[4]	CT[3]	CT[2]	CT[1]	CT[0]	ST[9]	ST[8]	ST[7]	ST[6]	ST[5]	ST[4]	ST[3]	ST[2]	ST[1]	ST[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0

Table 36-14. ADC Conversion Time Setting Register (ADCn_CT) bits

Bit Position	Bit Field Name	Bit Description
[15:10]	CT	A/D Comparison Time Set These bits specify the comparison time. Comparison time = Comparison time = (CT value * 10 + 4) * CLK_PERI0_PD2 cycle.
[9:0]	ST	Analog Input Sampling Time Set These bits specify the sampling time of the input. Sampling time = ST value X CLK_PERI0_PD2 cycle.

Note: The necessary sampling time and ST value are calculated using the formulas on the following page.

The sampling time should be set to a minimum of '7τ'. The following approximation formulas can be used for Figure 36-16:

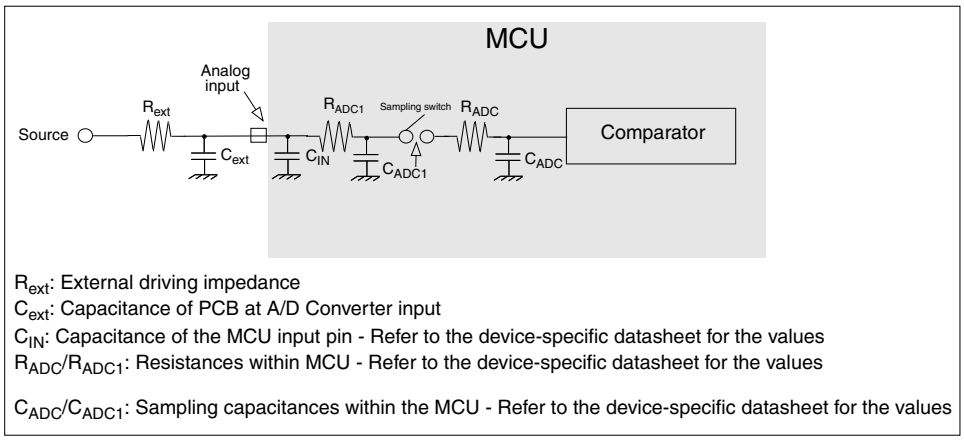
ADC_AN0..25:

$$T_{samp}[min] = 7.63 \times [R_{ext} \times (C_{ext} + C_{IN}) + (R_{ext} + R_{ADC}) \times C_{ADC}]$$

ADC_AN26..31:

$$T_{samp}[min] = 7.63 \times [R_{ext} \times (C_{ext} + C_{IN}) + (R_{ext} + R_{ADC1}) \times C_{ADC1} + (R_{ext} + R_{ADC1} + R_{ADC}) \times C_{ADC}]$$

Figure 36-16. Equivalent analog input circuit



R_{ext}: External driving impedance
C_{ext}: Capacitance of PCB at A/D Converter input
C_{IN}: Capacitance of the MCU input pin - Refer to the device-specific datasheet for the values
R_{ADC}/R_{ADC1}: Resistances within MCU - Refer to the device-specific datasheet for the values

C_{ADC}/C_{ADC1}: Sampling capacitances within the MCU - Refer to the device-specific datasheet for the values

If the sampling time is not sufficient, connect a capacitor of about 0.1 µF to the analog input pin. In this case the internal sampling capacitance CADC is charged out of this external capacitance.

A big external driving impedance also adversely affects the A/D conversion precision due to the pin input leakage current I_{IL} (static current before the sampling switch) or the analog input leakage current I_{AIN} (total leakage current of pin input and comparator during sampling). The effect of the pin input leakage current I_{IL} cannot be compensated by an external capacitor. The accuracy gets worse as |AVRH - AVRL| becomes smaller.

36.2.13 A/D Start Channel Setting Register (ADCn_SCH)

This register specifies the start channel for the A/D Converter to convert. S/W must not update this register while the A/D Converter is operating.

A/D Start Channel Setting Register (ADCn_SCH)

Figure 36-17. A/D Start Channel Setting Register (ADCn_SCH)

ADCn_SCH							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	ANS[4]	ANS[3]	ANS[2]	ANS[1]	ANS[0]
Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-15. A/D Start Channel Setting Register (ADCn_SCH) bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	ANS	Analog Start Channel Set These bits set the start channel for A/D Converter. In continuous, single mode, or stop mode conversions start from the channels specified by ANS bits and continues up to the channel specified by the ADCn_ECH:ANE bits. Conversion then starts again from the start channel specified by ANS bits. If ANS>ADCn_ECH:ANE, conversion starts with the channel specified by ANS, continues up to channel 31, starts again from channel 0 and ends with the channel specified by ADCn_ECH:ANE. As the channel disable function is valid in each case (not only if ANS>ADCn_ECH:ANE). The channels that are disabled by ADCn_ER32:ADE[15:0], ADCn_ER10:ADE[31:16] will be skipped. For the ADC to start conversion, the channel selected by ANS[4:0] bits (start channel) must be enabled by the corresponding ADCn_ER32:ADE[15:0], ADCn_ER10[31:16]:ADE bit.

36.2.14 A/D End Channel Setting Register (ADCn_ECH)

This register specifies the end channel for the A/D Converter to convert. S/W must not update this register while the A/D Converter is operating.

A/D End Channel Setting Register (ADCn_ECH)

Figure 36-18. A/D End Channel Setting Register (ADCn_ECH)

ADCn_ECH							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	ANE[4]	ANE[3]	ANE[2]	ANE[1]	ANE[0]
Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-16. A/D End Channel Setting Register (ADCn_ECH) bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	ANE	Analog End Channel Set These bits set the end channel for A/D Converter. Setting of ANE[4:0] the same channel as ADCn_SCH:ANS[4:0] specifies conversion for that channel only. In continuous, single, or stop mode, conversion is performed up to the channel specified by ANE[4:0].

Note: Example: Channel setting ADCn_SCH:ANS = 30, ADCn_ECH:ANE = 3, single conversion mode. Operation: Conversion channel 30 -> 31-> 0 -> 1 -> 2 -> 3 end.

Table 36-17. ADCn_SCH:ANS and ADCn_ECH:ANE bit configuration for start and end channel

ANS[4] ANE[4]	ANS[3] ANE[3]	ANS[2] ANE[2]	ANS[1] ANE[1]	ANS[0] ANE[0]	Start/end channel
'0'	'0'	'0'	'0'	'0'	AN0
'0'	'0'	'0'	'0'	'1'	AN1
'0'	'0'	'0'	'1'	'0'	AN2
'0'	'0'	'0'	'1'	'1'	AN3
		...			...
'1'	'1'	'1'	'0'	'1'	AN29
'1'	'1'	'1'	'1'	'0'	AN30
'1'	'1'	'1'	'1'	'1'	AN31

Channels that are disabled in ADCn_ER32:ADE[15:0], ADCn_ER10:ADE[31:16] in the range of ADCn_SCH:ANS and ADCn_ECH:ANE will be skipped.

36.2.15 A/D Converter DMA Configuration Register (ADCn_MAR)

This register contains bits to configure the DMA. It can enable/disable the DMA request from the A/D Converter.

A/D Converter DMA Configuration Register (ADCn_MAR)

Figure 36-19. A/D Converter DMA Configuration Register (ADCn_MAR)

ADCn_MAR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	DRQEN2	DRQEN
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-18. A/D Converter DMA Configuration Register (ADCn_MAR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	DRQEN2	DMA Request Enable for INT2 (end of scan) This bit enables the DMA request for end of scan interrupt. '0': DMA request is diabled '1': DMA request is enabled
[0]	DRQEN	DMA Request Enable for INT (end of conversion) This bit enables the DMA request for end of conversion interrupt. '0': DMA request is disabled '1': DMA request is ensabled Data protection is applicable only for DRQEN (end of conversion) if ADCn_CS2:DPDIS = '0'. When DRQEN = '1' and ADCn_CS2:DPDIS = '0' then the ADCn_CS1:INT (end of conversion interrupt flag) is automatically cleared after ADCn_CR register has been read (except by the debug master, DAP). The A/D conversion will be paused if the previous data was not read in time i.e. as long as ADCn_CS1:INT is '1'.

36.2.16 A/D Converter DMA Configuration Set Register (ADCn_MASR)

A/D Converter DMA Configuration Set Register (ADCn_MASR) contains bits to set the bits of the DMA Configuration Register (ADCn_MAR).

A/D Converter DMA Configuration Set Register (ADCn_MASR)

Figure 36-20. A/D Converter DMA Configuration Set Register (ADCn_MASR)

ADCn_MASR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	DRQENS2	DRQENS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 36-19. A/D Converter DMA Configuration Set Register (ADCn_MASR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	DRQENS2	DMA Enable for INT2 (end of scan) Set bit '0': No effect '1': Sets ADCn_MAR:DRQEN2 bit Reading this bit always returns '0'.
[0]	DRQENS	DMA Enable for INT (end of conversion) Set bit '0': No effect '1': Sets ADCn_MAR:DRQEN bit Reading this bit always returns '0'.

36.2.17 A/D Converter DMA Configuration Clear Register (ADCn_MACR)

A/D Converter DMA Configuration Clear Register (ADCn_MACR) contains bits to clear the bits of the DMA Configuration Register (ADCn_MAR).

A/D Converter DMA Configuration Clear Register (ADCn_MACR)

Figure 36-21. A/D Converter DMA Configuration Clear Register (ADCn_MACR)

ADCn_MACR							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	DRQENC2	DRQENC
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0

Table 36-20. A/D Converter DMA Configuration Clear Register (ADCn_MACR) bits

Bit Position	Bit Field Name	Bit Description
[7:2]	read0	-
[1]	DRQENC2	DMA Enable for INT2 (end of scan) Clear bit '0': No effect '1': Clears ADCn_MAR:DRQEN2 bit Reading this bit always returns '0'.
[0]	DRQENC	DMA Enable for INT (end of conversion) Clear bit '0': No effect '1': Clears ADCn_MAR:DRQEN bit Reading this bit always returns '0'.

36.2.18 Range Comparator Upper Threshold Registers (ADCn_RCOH3~0)

These registers specify the upper threshold values of the range comparator to which the output of the A/D Converter is compared. Only ADCn_RCOH0 for range comparator 0 is described here. Other registers (ADCn_RCOH1, ADCn_RCOH2, and ADCn_RCOH3,) have similar bit fields.

Range Comparator Upper Threshold Register 0 (ADCn_RCOH0)

Figure 36-22. Range Comparator Upper Threshold Register 0 (ADCn_RCOH0)

ADCn_RCOH0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOH[7]	RCOH[6]	RCOH[5]	RCOH[4]	RCOH[3]	RCOH[2]	RCOH[1]	RCOH[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	1	1	1	1

Table 36-21. Range Comparator Upper Threshold Register 0 (ADCn_RCOH0) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	RCOH	Range Comparator Upper Threshold The RCOH bits define the upper comparison threshold of the range comparator 0. The upper comparator compares the upper 8 bits of the ADC conversion result. If the value is higher than RCOH[7:0], then the conversion result is outside range.

36.2.19 Range Comparator Lower Threshold Registers (ADCn_RCOL3~0)

These registers specify the lower threshold values of the range comparator to which the output of the A/D Converter is compared. Only ADCn_RCOL0 for range comparator 0 is described here. Other registers (ADCn_RCOL1, ADCn_RCOL2, and ADCn_RCOL3) have similar bit fields.

Range Comparator Lower Threshold Register 0 (ADCn_RCOL0)

Figure 36-23. Range Comparator Lower Threshold Register 0 (ADCn_RCOL0)

ADCn_RCOL0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOL[7]	RCOL[6]	RCOL[5]	RCOL[4]	RCOL[3]	RCOL[2]	RCOL[1]	RCOL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-22. Range Comparator Lower Threshold Register 0 (ADCn_RCOL0) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	RCOL	Range Comparator Lower Threshold The RCOL bits define the lower comparison threshold of the range comparator 0. The lower comparator compares the upper 8 bits of the ADC conversion result. If the value is lower than RCOL[7:0], then the conversion result is outside range.

36.2.20 A/D Converter Channel Control Registers (ADCn_CC15~0)

The A/D Converter Channel Control Registers serve 2 ADC channels per register and control the range comparison for these channels. Only ADCn_CC0 is explained here. Other registers (ADCn_CC1, ADCn_CC2..... ADCn_CC15) have similar bit fields.

A/D Converter Channel Control Register 0 (ADCn_CC0)

Figure 36-24. A/D Converter Channel Control Register 0 (ADCn_CC0)

ADCn_CC0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOIE1	RCOE1	RCOS1[1]	RCOS1[0]	RCOIE0	RCOE0	RCOS0[1]	RCOS0[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-23. A/D Converter Channel Control Register 0 (ADCn_CC0) bits

Bit Position	Bit Field Name	Bit Description
[7]	RCOIE1	Enable the Range Comparator Interrupt for the Odd ADC Channel '0': RCO interrupt for this ADC channel is disabled '1': RCO interrupt for this ADC channel is enabled
[6]	RCOE1	Enable of Range Comparator for the Odd ADC Channel '0': RCO is disabled. RCO flags for this ADC channel will not be set '1': RCO is enabled
[5:4]	RCOS1	Range Comparator Selection for Odd ADC Channel '00': Select range comparator 0 for this ADC channel '01': Select range comparator 1 for this ADC channel '10': Select range comparator 2 for this ADC channel '11': Select range comparator 3 for this ADC channel
[3]	RCOIE0	Enable the Range Comparator Interrupt for the Even ADC Channel '0': RCO interrupt for this ADC channel is disabled '1': RCO interrupt for this ADC channel is enabled
[2]	RCOE0	Enable of Range Comparator for the Even ADC Channel '0': RCO is disabled. RCO flags for this ADC channel will not be set '1': RCO is enabled

Table 36-23. A/D Converter Channel Control Register 0 (ADCn_CC0) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	RCOS0	Range Comparator Selection for Even ADC Channel '00': Select range comparator 0 for this ADC channel '01': Select range comparator 1 for this ADC channel '10': Select range comparator 2 for this ADC channel '11': Select range comparator 3 for this ADC channel

Note:

Mapping of the ADC channels to the registers are done as follows:

ADCn_CC15~0[3:0] is mapped to even ADC input channels (0,2,4.....30).

ADCn_CC15~0[7:4] is mapped to odd ADC input channels (1,3,5.....31).

Example: Channel 5 will be mapped to ADCn_CC2[7:4] and channel 4 will be mapped to ADCn_CC2[3:0].

The result of the comparison can be filtered by the ADC pulse detection function. For further details on ADC pulse detection function and positive event/negative event refer to 36.3 Operation of A/D Converter.

36.2.21 Inverted Range Selection Registers (ADCn_RCOIRS32, ADCn_RCOIRS10)

The ADCn_RCOIRS32, ADCn_RCOIRS10 registers control the comparison check for 'outside range' or 'inside range'. The 32 bits of ADCn_RCOIRS are organized 'per ADC channel'. ADC channel 0 is located on the MSB of the register and ADC channel 31 is on the LSB.

Inverted Range Selection Register (ADCn_RCOIRS32)

Figure 36-25. Inverted Range Selection Register (ADCn_RCOIRS32)

ADCn_RCOIRS32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOIRS0	RCOIRS1	RCOIRS2	RCOIRS3	RCOIRS4	RCOIRS5	RCOIRS6	RCOIRS7	RCOIRS8	RCOIRS9	RCOIRS10	RCOIRS11	RCOIRS12	RCOIRS13	RCOIRS14	RCOIRS15
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-24. Inverted Range Selection Register (ADCn_RCOIRS32) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOIRS0	Inverted Range Selection for ADC Channel 0 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 0. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF0, ADCn_RCOINT32:RCOINT0) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called the 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT0) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called the 'inside range' mode

Table 36-24. Inverted Range Selection Register (ADCn_RCOIRS32) bits

Bit Position	Bit Field Name	Bit Description
[14]	RCOIRS1	Inverted Range Selection for ADC Channel 1 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 1. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF1, ADCn_RCOINT32:RCOINT1) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called the 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT1) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called the 'inside range' mode
[13]	RCOIRS2	Inverted Range Selection for ADC Channel 2 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 2. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF2, ADCn_RCOINT32:RCOINT2) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called the 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT2) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called the 'inside range' mode
[12]	RCOIRS3	Inverted Range Selection for ADC Channel 3 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 3. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF3, ADCn_RCOINT32:RCOINT3) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT3) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-24. Inverted Range Selection Register (ADCn_RCOIRS32) bits

Bit Position	Bit Field Name	Bit Description
[11]	RCOIRS4	Inverted Range Selection for ADC Channel 4 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 4. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF4, ADCn_RCOINT32:RCOINT4) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT4) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[10]	RCOIRS5	Inverted Range Selection for ADC Channel 5 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 5. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF5, ADCn_RCOINT32:RCOINT5) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT5) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[9]	RCOIRS6	Inverted Range Selection for ADC Channel 6 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 6. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF6, ADCn_RCOINT32:RCOINT6) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT6) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-24. Inverted Range Selection Register (ADCn_RCOIRS32) bits

Bit Position	Bit Field Name	Bit Description
[8]	RCOIRS7	Inverted Range Selection for ADC Channe 7 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 7. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF7, ADCn_RCOINT32:RCOINT7) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT7) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[7]	RCOIRS8	Inverted Range Selection for ADC Channel 8 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 8. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF8, ADCn_RCOINT32:RCOINT8) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT8) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[6]	RCOIRS9	Inverted Range Selection for ADC Channel 9 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 0. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF9, ADCn_RCOINT32:RCOINT9) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT9) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-24. Inverted Range Selection Register (ADCn_RCOIRS32) bits

Bit Position	Bit Field Name	Bit Description
[5]	RCOIRS10	Inverted Range Selection for ADC Channel 10 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 10. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF10, ADCn_RCOINT32:RCOINT10) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT10) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[4]	RCOIRS11	Inverted Range Selection for ADC Channel 11 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 11. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF11, ADCn_RCOINT32:RCOINT11) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT11) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[3]	RCOIRS12	Inverted Range Selection for ADC Channel 12 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 12. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF12, ADCn_RCOINT32:RCOINT12) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT12) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-24. Inverted Range Selection Register (ADCn_RCOIRS32) bits

Bit Position	Bit Field Name	Bit Description
[2]	RCOIRS13	Inverted Range Selection for ADC Channel 13 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 13. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF13, ADCn_RCOINT32:RCOINT13) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT13) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[1]	RCOIRS14	Inverted Range Selection for ADC Channel 14 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 14. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF14, ADCn_RCOINT32:RCOINT14) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT14) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[0]	RCOIRS15	Inverted Range Selection for ADC Channel 15 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 15. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF32:RCOOF15, ADCn_RCOINT32:RCOINT15) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT32:RCOINT15) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Inverted Range Selection Register (ADCn_RCOIRS10)

Figure 36-26. Inverted Range Selection Register (ADCn_RCOIRS10) bits

ADCn_RCOIRS10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOIRS16	RCOIRS17	RCOIRS18	RCOIRS19	RCOIRS20	RCOIRS21	RCOIRS22	RCOIRS23	RCOIRS24	RCOIRS25	RCOIRS26	RCOIRS27	RCOIRS28	RCOIRS29	RCOIRS30	RCOIRS31
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-25. Inverted Range Selection Register (ADCn_RCOIRS10) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOIRS16	Inverted Range Selection for ADC Channel 16 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 16. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF16, ADCn_RCOINT10:RCOINT16) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called the 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT16) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called the 'inside range' mode

Table 36-25. Inverted Range Selection Register (ADCn_RCOIRS10) bits

Bit Position	Bit Field Name	Bit Description
[14]	RCOIRS17	Inverted Range Selection for ADC Channel 17 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 17. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF17, ADCn_RCOINT10:RCOINT17) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT17) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[13]	RCOIRS18	Inverted Range Selection for ADC Channel 18 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 18. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF18, ADCn_RCOINT10:RCOINT18) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT18) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[12]	RCOIRS19	Inverted Range Selection for ADC Channel 19 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 19. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_R-COOF10RCOOF10:RCOOF19, ADCn_RCOINT10:RCOINT19) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT19) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-25. Inverted Range Selection Register (ADCn_RCOIRS10) bits

Bit Position	Bit Field Name	Bit Description
[11]	RCOIRS20	Inverted Range Selection for ADC Channel 20 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 20. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF20, ADCn_RCOINT10:RCOINT20) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT20) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[10]	RCOIRS21	Inverted Range Selection for ADC Channel 21 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 21. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF21, ADCn_RCOINT10:RCOINT21) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT21) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[9]	RCOIRS22	Inverted Range Selection This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 22. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF22, ADCn_RCOINT10:RCOINT22) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT22) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-25. Inverted Range Selection Register (ADCn_RCOIRS10) bits

Bit Position	Bit Field Name	Bit Description
[8]	RCOIRS23	Inverted Range Selection This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 23. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF23, ADCn_RCOINT10:RCOINT23) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT23) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[7]	RCOIRS24	Inverted Range Selection This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 24. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF24, ADCn_RCOINT10:RCOINT24) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT24) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[6]	RCOIRS25	Inverted Range Selection for ADC Channel 25 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 25. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF25, ADCn_RCOINT10:RCOINT25) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT25) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-25. Inverted Range Selection Register (ADCn_RCOIRS10) bits

Bit Position	Bit Field Name	Bit Description
[5]	RCOIRS26	Inverted Range Selection for ADC Channel 26 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 26. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF26, ADCn_RCOINT10:RCOINT26) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT26) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[4]	RCOIRS27	Inverted Range Selection for ADC Channel 27 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 27. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF27, ADCn_RCOINT10:RCOINT27) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT27) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[3]	RCOIRS28	Inverted Range Selection for ADC Channel 28 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 28. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF28, ADCn_RCOINT10:RCOINT28) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Table 36-25. Inverted Range Selection Register (ADCn_RCOIRS10) bits

Bit Position	Bit Field Name	Bit Description
[2]	RCOIRS29	Inverted Range Selection for ADC Channel 29 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 29. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF29, ADCn_RCOINT10:RCOINT29) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT29) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[1]	RCOIRS30	Inverted Range Selection for ADC Channel 30 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 30. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF30, ADCn_RCOINT10:RCOINT30) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT30) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode
[0]	RCOIRS31	Inverted Range Selection for ADC Channel 31 This bit controls how the range comparator result flags are set. This bit is assigned to ADC channel 31. '0': The comparison checks for 'outside range' i.e., the over threshold and interrupt flags (ADCn_RCOOF10:RCOOF31, ADCn_RCOINT10:RCOINT31) are set when the ADC result is above the upper threshold OR below the lower threshold. That is called 'outside range' mode '1': The comparison checks for 'inside range' i.e., the interrupt flags (ADCn_RCOINT10:RCOINT31) are set when the ADC result is below or equal the upper threshold and above or equal the lower threshold. That is called 'inside range' mode

Note: All the bits in the registers ADCn_RCOIRS32 and ADCn_RCOIRS10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.2.22 Range Comparator Interrupt Flags (ADCn_RCOINT32, ADCn_RCOINT10)

The result of range comparison are stored in two flag registers, Range Comparator Interrupt Flags (ADCn_RCOINT32, ADCn_RCOINT10) and Range Comparator Over Threshold Flags (ADCn_RCOOF32, ADCn_RCOOF10).

Range Comparator Interrupt Flag (ADCn_RCOINT32)

Figure 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT32)

ADCn_RCOINT32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOINT0	RCOINT1	RCOINT2	RCOINT3	RCOINT4	RCOINT5	RCOINT6	RCOINT7	RCOINT8	RCOINT9	RCOINT10	RCOINT11	RCOINT12	RCOIN13	RCOINT14	RCOINT15
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-26. Range Comparator Interrupt Flag (ADCn_RCOINT32) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOINT0	Range Comparator Interrupt Flags for ADC Channel 0 This flag shows that an outside range or inside range condition has been found on ADC channel 0. These bits are set under the following conditions: ADC channel 0 is enabled (ADCn_ER32:ADE0 = '1') and The range comparison for this channel is enabled (ADCn_CC0:RCOE0) is set The conversion of ADC channel 0 has just finished and An interrupt condition was found on ADC channel 0 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC0 bits.

Table 36-26. Range Comparator Interrupt Flag (ADCn_RCOINT32) bits

Bit Position	Bit Field Name	Bit Description
[14]	RCOINT1	Range Comparator Interrupt Flags for ADC Channel 1 This flag shows that an outside range or inside range condition has been found on the ADC channel 1. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE1 = '1' and The range comparison for this channel is enabled ADCn_CC0:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 1 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC1 bits.
[13]	RCOINT2	Range Comparator Interrupt Flags for ADC Channel 2 This flag shows that an outside range or inside range condition has been found on the ADC channel 2. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE2 = '1' and The range comparison for this channel is enabled ADCn_CC1:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 2 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC2 bits.
[12]	RCOINT3	Range Comparator Interrupt Flags for ADC Channel 3 This flag shows that an outside range or inside range condition has been found on the ADC channel 3. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE3 = '1' and The range comparison for this channel is enabled ADCn_CC1:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 3 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC3 bits.

Table 36-26. Range Comparator Interrupt Flag (ADCn_RCOINT32) bits

Bit Position	Bit Field Name	Bit Description
[11]	RCOINT4	Range Comparator Interrupt Flags for ADC Channel 4 This flag shows that an outside range or inside range condition has been found on the ADC channel 4. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE4 = '1' and The range comparison for this channel is enabled ADCn_CC2:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 4 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC4 bits.
[10]	RCOINT5	Range Comparator Interrupt Flags for ADC Channel 5 This flag shows that an outside range or inside range condition has been found on the ADC channel 5. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE5 = '1' and The range comparison for this channel is enabled ADCn_CC2:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 5(see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC5 bits.
[9]	RCOINT6	Range Comparator Interrupt Flags for ADC Channel 6 This flag shows that an outside range or inside range condition has been found on the ADC channel 6. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE6 = '1' and The range comparison for this channel is enabled ADCn_CC3:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 6 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC6 bits.

Table 36-26. Range Comparator Interrupt Flag (ADCn_RCOINT32) bits

Bit Position	Bit Field Name	Bit Description
[8]	RCOINT7	Range Comparator Interrupt Flags for ADC Channel 7 This flag shows that an outside range or inside range condition has been found on the ADC channel 7. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE7 = '1' and The range comparison for this channel is enabled ADCn_CC3:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 7 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC7 bits.
[7]	RCOINT8	Range Comparator Interrupt Flags for ADC Channel 8 This flag shows that an outside range or inside range condition has been found on the ADC channel 8. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE8 = '1' and The range comparison for this channel is enabled ADCn_CC4:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 8 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC8 bits.
[6]	RCOINT9	Range Comparator Interrupt Flags for ADC Channel 9 This flag shows that an outside range or inside range condition has been found on the ADC channel 9. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE9 = '1' and The range comparison for this channel is enabled ADCn_CC4:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 9 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC9 bits.

Table 36-26. Range Comparator Interrupt Flag (ADCn_RCOINT32) bits

Bit Position	Bit Field Name	Bit Description
[5]	RCOINT10	Range Comparator Interrupt Flags for ADC Channel 10 This flag shows that an outside range or inside range condition has been found on the ADC channel 10. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE10 = '1' and The range comparison for this channel is enabled ADCn_CC5:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 10 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC10 bits.
[4]	RCOINT11	Range Comparator Interrupt Flags for ADC Channel 11 This flag shows that an outside range or inside range condition has been found on the ADC channel 11. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE11 = '1' and The range comparison for this channel is enabled ADCn_CC5:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 11 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC11 bits.
[3]	RCOINT12	Range Comparator Interrupt Flags for ADC Channel 12 This flag shows that an outside range or inside range condition has been found on the ADC channel 12. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE12 = '1' and The range comparison for this channel is enabled ADCn_CC6:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 12 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC12 bits.

Table 36-26. Range Comparator Interrupt Flag (ADCn_RCOINT32) bits

Bit Position	Bit Field Name	Bit Description
[2]	RCOIN13	Range Comparator Interrupt Flags for ADC Channel 13 This flag shows that an outside range or inside range condition has been found on the ADC channel 13. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE13 = '1' and The range comparison for this channel is enabled ADCn_CC6:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 13(see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC13 bits.
[1]	RCOINT14	Range Comparator Interrupt Flags for ADC Channel 14 This flag shows that an outside range or inside range condition has been found on the ADC channel 14. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE14 = '1' and The range comparison for this channel is enabled ADCn_CC7:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 14 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC14 bits.
[0]	RCOINT15	Range Comparator Interrupt Flags for ADC Channel 15 This flag shows that an outside range or inside range condition has been found on the ADC channel 15. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER32:ADE15 = '1' and The range comparison for this channel is enabled ADCn_CC7:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 15 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC32:RCOINTC15 bits.

Range Comparator Interrupt Flag (ADCn_RCOINT10)

Figure 36-28. Range Comparator Interrupt Flag (ADCn_RCOINT10)

ADCn_RCOINT10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOINT16	RCOINT17	RCOINT18	RCOINT19	RCOINT20	RCOINT21	RCOINT22	RCOINT23	RCOINT24	RCOINT25	RCOINT26	RCOINT27	RCOINT28	RCOINT29	RCOINT30	RCOINT31
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT10) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOINT16	Range Comparator Interrupt Flags for ADC Channel 16 This flag shows that an outside range or inside range condition has been found on ADC channel 16. These bits are set under the following conditions: ADC channel 16 is enabled (ADCn_ER10:ADE16 = '1') The range comparison for this channel is enabled (ADCn_CC8:RCOE0) is set The conversion of ADC channel 16 has just finished An interrupt condition was found on ADC channel 16 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC16 bits.
[14]	RCOINT17	Range Comparator Interrupt Flags for ADC Channel 17 This flag shows that an outside range or inside range condition has been found on the ADC channel 17. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE17 = '1' and The range comparison for this channel is enabled ADCn_CC8:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 17 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC17 bits.

Table 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT10) bits

Bit Position	Bit Field Name	Bit Description
[13]	RCOINT18	Range Comparator Interrupt Flags for ADC Channel 18 This flag shows that an outside range or inside range condition has been found on the ADC channel 18. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE18 = '1'and The range comparison for this channel is enabled ADCn_CC9:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 18 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC18 bits.
[12]	RCOINT19	Range Comparator Interrupt Flags for ADC Channel 19 This flag shows that an outside range or inside range condition has been found on the ADC channel 19. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE19 = '1' and The range comparison for this channel is enabled ADCn_CC9:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 19(see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC19 bits.
[11]	RCOINT20	Range Comparator Interrupt Flags for ADC Channel 20 This flag shows that an outside range or inside range condition has been found on the ADC channel 20. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE20= '1' and The range comparison for this channel is enabled ADCn_CC10:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 20 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC20 bits.

Table 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT10) bits

Bit Position	Bit Field Name	Bit Description
[10]	RCOINT21	Range Comparator Interrupt Flags for ADC Channel 21 This flag shows that an outside range or inside range condition has been found on the ADC channel 21. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE21 = '1'and The range comparison for this channel is enabled ADCn_CC10:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 21 (see Table 36-28). This bit is cleared by writing '1' to the correspond-ing ADCn_RCOINTC10:RCOINTC21 bits.
[9]	RCOINT22	Range Comparator Interrupt Flags for ADC Channel 22 This flag shows that an outside range or inside range condition has been found on the ADC channel 22. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE22 = '1' and The range comparison for this channel is enabled ADCn_CC11:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 22 (see Table 36-28). This bit is cleared by writing '1' to the correspond-ing ADCn_RCOINTC10:RCOINTC22 bits.
[8]	RCOINT23	Range Comparator Interrupt Flags for ADC Channel 23 This flag shows that an outside range or inside range condition has been found on the ADC channel 23. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE23 = '1' and The range comparison for this channel is enabled ADCn_CC11:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 23 (see Table 36-28). This bit is cleared by writing '1' to the correspond-ing ADCn_RCOINTC10:RCOINTC23 bits.

Table 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT10) bits

Bit Position	Bit Field Name	Bit Description
[7]	RCOINT24	Range Comparator Interrupt Flags for ADC Channel 24 This flag shows that an outside range or inside range condition has been found on the ADC channel 24. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE24 = '1' and The range comparison for this channel is enabled ADCn_CC12:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 24 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC24 bits.
[6]	RCOINT25	Range Comparator Interrupt Flags for ADC Channel 25 This flag shows that an outside range or inside range condition has been found on the ADC channel 25. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE25 = '1' and The range comparison for this channel is enabled ADCn_CC12:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 25 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC25 bits.
[5]	RCOINT26	Range Comparator Interrupt Flags for ADC Channel 26 This flag shows that an outside range or inside range condition has been found on the ADC channel 26. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE26 = '1'and The range comparison for this channel is enabled ADCn_C13:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 26 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC26 bits.

Table 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT10) bits

Bit Position	Bit Field Name	Bit Description
[4]	RCOINT27	Range Comparator Interrupt Flags for ADC Channel 27 This flag shows that an outside range or inside range condition has been found on the ADC channel 27. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE27 = '1' and The range comparison for this channel is enabled ADCn_CC13:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 27 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC27 bits.
[3]	RCOINT28	Range Comparator Interrupt Flags for ADC Channel 28 This flag shows that an outside range or inside range condition has been found on the ADC channel 28. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE28 = '1' and The range comparison for this channel is enabled ADCn_CC14:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 28 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC28 bits.
[2]	RCOINT29	Range Comparator Interrupt Flags for ADC Channel 29 This flag shows that an outside range or inside range condition has been found on the ADC channel 29. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE29 = '1' and The range comparison for this channel is enabled ADCn_CC14:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 29 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC29 bits.

Table 36-27. Range Comparator Interrupt Flag (ADCn_RCOINT10) bits

Bit Position	Bit Field Name	Bit Description
[1]	RCOINT30	Range Comparator Interrupt Flags for ADC Channel 30 This flag shows that an outside range or inside range condition has been found on the ADC channel 30. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE30 = '1' and The range comparison for this channel is enabled ADCn_CC15:RCOE0 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 30 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC30 bits.
[0]	RCOINT31	Range Comparator Interrupt Flags for ADC Channel 31 This flag shows that an outside range or inside range condition has been found on the ADC channel 31. These bits are set under the following conditions: The ADC channel is enabled ADCn_ER10:ADE31 = '1' and The range comparison for this channel is enabled ADCn_CC15:RCOE1 is set and The conversion of the ADC channel is just finished and An interrupt condition was found on ADC channel 31 (see Table 36-28). This bit is cleared by writing '1' to the corresponding ADCn_RCOINTC10:RCOINTC31 bits.

Note: All the bits in the registers ADCn_RCOINT32 and ADCn_RCOINT10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

Table 36-28. Range comparator interrupt condition

Mode	ADCn_RCOIRS32: RCOIRS	Conversion result above upper threshold	Conversion result below lower threshold	Interrupt condition
Outside range	'0'	'1'	x	INT condition: above range, ADCn_RCOOF32, ADCn_RCOOF10:RCOOF is set.
		'0'	'0'	-
		'0'	'1'	INT condition: below range, ADCn_RCOOF32, ADCn_RCOOF10:RCOOF is cleared.

Table 36-28. Range comparator interrupt condition

Mode	ADCn_RCOIRS32: RCOIRS	Conversion result above upper threshold	Conversion result below lower threshold	Interrupt condition
Inside range	'1'	'1'	x	-
		'0'	'0'	INT condition: inside range.
		x	'1'	-

36.2.23 Range Comparator Over Threshold Flags (ADCn_RCOOF32, ADCn_RCOOF10)

The (ADCn_RCOOF32, ADCn_RCOOF10) flag bits are set when the result of the range comparator is 'outside range' and the converted value is above the value of the upper threshold register.

Range Comparator Over Threshold Flag (ADCn_RCOOF32)

Figure 36-29. Range Comparator Over Threshold Flag (ADCn_RCOOF32)

ADCn_RCOOF32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOOF0	RCOOF1	RCOOF2	RCOOF3	RCOOF4	RCOOF5	RCOOF6	RCOOF7	RCOOF8	RCOOF9	RCOOF10	RCOOF11	RCOOF12	RCOOF13	RCOOF14	RCOOF15
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-29. Range Comparator Over Threshold Flag (ADCn_RCOOF32) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOOF0	Range Comparator Over Threshold Flag for ADC Channel 0 This flag is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS0 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT0 = '1'), this flag has the following meaning: '0': The conversion result of the ADC channel 0 is less than or equal to the upper threshold '1': The conversion result of the ADC channel 0 is above the upper threshold
[14]	RCOOF1	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 1. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS1 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT1 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 1 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 1 is above the upper threshold

Table 36-29. Range Comparator Over Threshold Flag (ADCn_RCOOF32) bits

Bit Position	Bit Field Name	Bit Description
[13]	RCOOF2	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 2. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS2 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT2 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 2 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 2 is above the upper threshold
[12]	RCOOF3	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 3. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS3 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT3 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 3 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 3 is above the upper threshold
[11]	RCOOF4	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 4. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS4 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT4 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 4 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 4 is above the upper threshold

Table 36-29. Range Comparator Over Threshold Flag (ADCn_RCOOF32) bits

Bit Position	Bit Field Name	Bit Description
[10]	RCOOF5	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 5. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS5 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT5 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 5 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 5 is above the upper threshold
[9]	RCOOF6	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 6. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS6 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT6 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 6 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 6 is above the upper threshold
[8]	RCOOF7	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 7. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS7 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT7 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 7 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 7 is above the upper threshold

Table 36-29. Range Comparator Over Threshold Flag (ADCn_RCOOF32) bits

Bit Position	Bit Field Name	Bit Description
[7]	RCOOF8	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 8. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS8 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT8 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 8 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 8 is above the upper threshold
[6]	RCOOF9	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 9. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS9 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT9 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 9 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 9 is above the upper threshold
[5]	RCOOF10	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 10. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS10 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT10 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 10 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 10 is above the upper threshold
[4]	RCOOF11	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 11. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS11 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT11 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 11 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 11 is above the upper threshold

Table 36-29. Range Comparator Over Threshold Flag (ADCn_RCOOF32) bits

Bit Position	Bit Field Name	Bit Description
[3]	RCOOF12	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 12. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS12 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT12 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 12 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 12 is above the upper threshold
[2]	RCOOF13	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 13. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS13 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT13 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 13 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 13 is above the upper threshold
[1]	RCOOF14	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS14 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT14 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 14 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 14 is above the upper threshold
[0]	RCOOF15	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 15. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS32:RCOIRS15 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT32:RCOINT15 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 15 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 15 is above the upper threshold

Range Comparator Over Threshold Flag (ADCn_RCOOF10)

Figure 36-30. Range Comparator Over Threshold Flag (ADCn_RCOOF10)

ADCn_RCOOF10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOOF16	RCOOF17	RCOOF18	RCOOF19	RCOOF20	RCOOF21	RCOOF22	RCOOF23	RCOOF24	RCOOF25	RCOOF26	RCOOF27	RCOOF28	RCOOF29	RCOOF30	RCOOF31
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-30. Range Comparator Over Threshold Flag (ADCn_RCOOF10) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOOF16	Range Comparator Over Threshold Flag for ADC Channel 16 The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS16 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT16 = '1'), this flag has the following meaning: '0': The conversion result of channel 16 is less than or equal to the upper threshold '1': The conversion result of channel 16 is above the upper threshold
[14]	RCOOF17	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 17. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS17 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT17 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 17 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 17 is above the upper threshold

Table 36-30. Range Comparator Over Threshold Flag (ADCn_RCOOF10) bits

Bit Position	Bit Field Name	Bit Description
[13]	RCOOF18	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 18. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS18 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT18 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 18 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 18 is above the upper threshold
[12]	RCOOF19	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 19. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS19 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT19 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 19 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 19 is above the upper threshold
[11]	RCOOF20	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 20. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS20 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT20 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 20 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 20 is above the upper threshold
[10]	RCOOF21	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 21. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS21 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT21 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 21 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 21 is above the upper threshold

Table 36-30. Range Comparator Over Threshold Flag (ADCn_RCOOF10) bits

Bit Position	Bit Field Name	Bit Description
[9]	RCOOF22	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 22. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS22 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT22 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 22 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 22 is above the upper threshold
[8]	RCOOF23	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 23. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS23 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT23 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 23 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 23 is above the upper threshold
[7]	RCOOF24	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 24. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS24 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT24 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 24 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 24 is above the upper threshold
[6]	RCOOF25	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 25. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS25 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT25 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 25 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 25 is above the upper threshold

Table 36-30. Range Comparator Over Threshold Flag (ADCn_RCOOF10) bits

Bit Position	Bit Field Name	Bit Description
[5]	RCOOF26	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 26. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS26 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT26 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 26 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 26 is above the upper threshold
[4]	RCOOF27	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 27. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS27 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT27 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 27 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 27 is above the upper threshold
[3]	RCOOF28	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 28. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS28 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT28 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 28 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 28 is above the upper threshold
[2]	RCOOF29	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 29. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS29 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT29 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 29 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 29 is above the upper threshold
[1]	read0	-

Table 36-30. Range Comparator Over Threshold Flag (ADCn_RCOOF10) bits

Bit Position	Bit Field Name	Bit Description
[0]	RCOOF31	Range Comparator Over Threshold Flag Each bit corresponds to one ADC channel 31. The flag of a channel is only applicable in 'outside range' mode i.e. while the corresponding bit ADCn_RCOIRS10:RCOIRS31 = '0'. If a comparison interrupt is signaled (corresponding bit ADCn_RCOINT10:RCOINT31 = '1'), this flag has the following meaning: '0': The conversion result of the corresponding channel 31 is less than or equal to the upper threshold '1': The conversion result of the corresponding channel 31 is above the upper threshold

Notes:

1. All the bits in the registers ADCn_RCOOF32 and ADCn_RCOOF10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.
2. The ADCn_RCOOF32, ADCn_RCOOF10:RCOOF flags for an ADC channel are loaded with the upper threshold comparator output signal under the following condition:
   - The corresponding ADCn_RCOIRS32, ADCn_RCOIRS10:RCOIRS flag is '0', i.e. configured for outside range and
   - The corresponding interrupt flag ADCn_RCOINT32, ADCn_RCOINT10:RCOINT has a rising edge
3. In this ADCn_RCOOF32 register bit[15] is assigned to ADC channel 0, ADCn_RCOOF10 register bit[0] is assigned to ADC channel 31.

36.2.24 Range Comparator Interrupt Clear Registers (ADCn_RCOINTC32, ADCn_RCOINTC10)

The Range Comparator Interrupt Clear Registers (ADCn_RCOINTC32, ADCn_RCOINTC10) contain bits to clear the bits of Range Comparator Interrupt Registers (ADCn_RCOINT32, ADCn_RCOINT10).

Range Comparator Interrupt Clear Register (ADCn_RCOINTC32)

Figure 36-31. Range Comparator Interrupt Clear Register (ADCn_RCOINTC32)

ADCn_RCOINTC32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOINTC0	RCOINTC1	RCOINTC2	RCOINTC3	RCOINTC4	RCOINTC5	RCOINTC6	RCOINTC7	RCOINTC8	RCOINTC9	RCOINTC10	RCOINTC11	RCOINTC12	RCOINTC13	RCOINTC14	RCOINTC15
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-31. Range Comparator Interrupt Clear Register (ADCn_RCOINTC32) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOINTC0	Range Comparator Interrupt Clear bit for ADC Channel 0 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT0 bit Reading this bit always returns '0'.
[14]	RCOINTC1	Range Comparator Interrupt Clear bit for ADC Channel 1 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT1 bit Reading this bit always returns '0'.
[13]	RCOINTC2	Range Comparator Interrupt Clear bit for ADC Channel 2 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT2 bit Reading this bit always returns '0'.
[12]	RCOINTC3	Range Comparator Interrupt Clear bit for ADC Channel 3 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT3 bit Reading this bit always returns '0'.

Table 36-31. Range Comparator Interrupt Clear Register (ADCn_RCOINTC32) bits

Bit Position	Bit Field Name	Bit Description
[11]	RCOINTC4	Range Comparator Interrupt Clear bit for ADC Channel 4 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT4 bit Reading this bit always returns '0'.
[10]	RCOINTC5	Range Comparator Interrupt Clear bit for ADC Channel 5 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT5 bit Reading this bit always returns '0'.
[9]	RCOINTC6	Range Comparator Interrupt Clear bit for ADC Channel 6 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT6 bit Reading this bit always returns '0'.
[8]	RCOINTC7	Range Comparator Interrupt Clear bit '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT7 bit Reading this bit always returns '0'.
[7]	RCOINTC8	Range Comparator Interrupt Clear bit for ADC Channel 8 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT8 bit Reading this bit always returns '0'.
[6]	RCOINTC9	Range Comparator Interrupt Clear bit for ADC Channel 9 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT9 bit Reading this bit always returns '0'.
[5]	RCOINTC10	Range Comparator Interrupt Clear bit for ADC Channel 10 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT10 bit Reading this bit always returns '0'.
[4]	RCOINTC11	Range Comparator Interrupt Clear bit for ADC Channel 11 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT11 bit Reading this bit always returns '0'.
[3]	RCOINTC12	Range Comparator Interrupt Clear bit for ADC Channel 12 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT12 bit Reading this bit always returns '0'.

Table 36-31. Range Comparator Interrupt Clear Register (ADCn_RCOINTC32) bits

Bit Position	Bit Field Name	Bit Description
[2]	RCOINTC13	Range Comparator Interrupt Clear bit for ADC Channel 13 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT13 bit Reading this bit always returns '0'.
[1]	RCOINTC14	Range Comparator Interrupt Clear bit for ADC Channel 14 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT14 bit Reading this bit always returns '0'.
[0]	RCOINTC15	Range Comparator Interrupt Clear bit for ADC Channel 15 '0': No effect '1': Clears corresponding ADCn_RCOINT32:RCOINT15 bit Reading this bit always returns '0'.

Range Comparator Interrupt Clear Register (ADCn_RCOINTC10)

Figure 36-32. Range Comparator Interrupt Clear Register (ADCn_RCOINTC10)

ADCn_RCOINTC10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
RCOINTC16	RCOINTC17	RCOINTC18	RCOINTC19	RCOINTC20	RCOINTC21	RCOINTC22	RCOINTC23	RCOINTC24	RCOINTC25	RCOINTC26	RCOINTC27	RCOINTC28	RCOINTC29	RCOINTC30	RCOINTC31
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-32. Range Comparator Interrupt Clear Register (ADCn_RCOINTC10) bits

Bit Position	Bit Field Name	Bit Description
[15]	RCOINTC16	Range Comparator Interrupt Clear bit for ADC channel 16 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT16 bit Reading this bit always returns '0'.
[14]	RCOINTC17	Range Comparator Interrupt Clear bit for ADC Channel 17 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT17 bit Reading this bit always returns '0'.

Table 36-32. Range Comparator Interrupt Clear Register (ADCn_RCOINTC10) bits

Bit Position	Bit Field Name	Bit Description
[13]	RCOINTC18	Range Comparator Interrupt Clear bit for ADC Channel 18 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT18 bit Reading this bit always returns '0'.
[12]	RCOINTC19	Range Comparator Interrupt Clear bit for ADC Channel 19 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT19 bit Reading this bit always returns '0'.
[11]	RCOINTC20	Range Comparator Interrupt Clear bit for ADC Channel 20 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT20 bit Reading this bit always returns '0'.
[10]	RCOINTC21	Range Comparator Interrupt Clear bit for ADC Channel 21 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT21 bit Reading this bit always returns '0'.
[9]	RCOINTC22	Range Comparator Interrupt Clear bit for ADC Channel 22 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT22 bit Reading this bit always returns '0'.
[8]	RCOINTC23	Range Comparator Interrupt Clear bit for ADC Channel 23 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT23 bit Reading this bit always returns '0'.
[7]	RCOINTC24	Range Comparator Interrupt Clear bit for ADC Channel 24 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT24 bit Reading this bit always returns '0'.
[6]	RCOINTC25	Range Comparator Interrupt Clear bit for ADC Channel 25 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT25 bit Reading this bit always returns '0'.
[5]	RCOINTC26	Range Comparator Interrupt Clear bit for ADC Channel 26 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT26 bit Reading this bit always returns '0'.

Table 36-32. Range Comparator Interrupt Clear Register (ADCn_RCOINTC10) bits

Bit Position	Bit Field Name	Bit Description
[4]	RCOINTC27	Range Comparator Interrupt Clear bit for ADC Channel 27 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT27 bit Reading this bit always returns '0'.
[3]	RCOINTC28	Range Comparator Interrupt Clear bit for ADC Channel 28 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT28 bit Reading this bit always returns '0'.
[2]	RCOINTC29	Range Comparator Interrupt Clear bit for ADC Channel 29 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT29 bit Reading this bit always returns '0'.
[1]	RCOINTC30	Range Comparator Interrupt Clear bit for ADC Channel 30 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT30 bit Reading this bit always returns '0'.
[0]	RCOINTC31	Range Comparator Interrupt Clear bit for ADC Channel 31 '0': No effect '1': Clears corresponding ADCn_RCOINT10:RCOINT31 bit Reading this bit always returns '0'.

Notes:

1. All the bits in the registers ADCn_RCOINTC32 and ADCn_RCOINTC10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.
2. Range comparator interrupt request generation:

The range comparator has a combined interrupt output line RCOIRQ. The interrupt output line becomes active if at least one of the range comparator interrupt flags ADCn_RCOINT32, ADCn_RCOINT10:RCOINT is set and the corresponding interrupt enable bit ADCn_CC15~0:RCOIE is set. It is not possible to activate a DMA request from the range comparator interrupts.

36.2.25 ADC Pulse Counter Reload Registers (ADCn_PCTNRL31~0/ADCn_PCTPRL31~0)

The ADC Pulse Counter Reload Registers hold the initial value of positive counters and negative counters of the ADC pulse detection function. The positive counters and negative counters are used for counting positive events and negative events of the range comparator. Each ADC channel has its own ADC Pulse Counter Reload Register for the positive and negative counter. Only ADCn_PCTNRL0 and ADCn_PCTPRL0 are described here. Other registers (ADCn_PCTNRL1, ADCn_PCTNRL2,....., ADCn_PCTNRL31 and ADCn_PCTPRL1, ADCn_PCTPRL2,....., ADCn_PCTPRL31) have similar bit fields.

ADC Pulse Negative Counter Reload Register 0 (ADCn_PCTNRL0)

Figure 36-33. ADC Pulse Negative Counter Reload Register 0 (ADCn_PCTNRL0)

ADCn_PCTNRL0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	D[4]	D[3]	D[2]	D[1]	D[0]
Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-33. ADC Pulse Negative Counter Reload Register 0 (ADCn_PCTNRL0) bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	D	ADC Pulse Negative Counter Reload Register These register bits hold the start value of the negative counter ADCn_PCTNCT0 used for counting negative events of ADC channel 0. The negative counter is reloaded with the value from this register when '1' is written to the corresponding ADCn_PCZFC32:CTPZFC,ADCn_PCZFC10:CTPZFC bit or on any positive event from the appropriate range comparator. For further explanation of negative events and operation of ADC pulse detection function refer to 'operation of ADC pulse detection function' in ADC pulse detection function.

Note: If the value of the Negative Counter Reload Register ADCn_PCTNRL31~0 is set to 0x00, the negative counter is stopped and filtering of negative events for the appropriate ADC channel is disabled.

ADC Pulse Positive Counter Reload Register 0 (ADCn_PCTPRL0)

Figure 36-34. ADC Pulse Positive Counter Reload Register 0 (ADCn_PCTPRL0)

ADCn_PCTPRL0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 36-34. ADC Pulse Positive Counter Reload Register 0 (ADCn_PCTPRL0) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	D	ADC Pulse Positive Counter Reload Register These register bits hold the start value of the positive counter ADCn_PCTPCT0 used for counting positive events of ADC channel 0. The positive counter is reloaded with the value from this register when '1' is written to the corresponding ADCn_PCZFC32:CTPZFC,ADCn_PCZFC10:CTPZFC bit or on expiration of the corresponding negative counter (ADCn_PCTNCT0). For further explanation of positive events and operation of ADC pulse detection function refer to 'operation of ADC pulse detection function' in ADC pulse detection function.

Note: If value of the Positive Counter Reload Register ADCn_PCTPRL0 is set to 0x00, the positive counter is stopped and the pulse detection function for the appropriate ADC channel is disabled.

36.2.26 ADC Pulse Counters (ADCn_PCTNCT31~0/ADCn_PCTPCT31~0)

These registers return the value of positive counter and negative counter of the ADC pulse detection function. ADCn_PCTNCT31~0, which is the negative counter, counts down the number of negative events and ADCn_PCTPCT31~0, which is the positive counter, counts down the number of positive events. Each ADC channel has its own positive and negative counter and the corresponding register. Only ADCn_PCTNCT0 and ADCn_PCTPCT0 are described here. Other registers (ADCn_PCTNCT1,....., ADCn_PCTNCT31 and ADCn_PCTPCT1,...., ADCn_PCTPCT31) have similar bit fields.

ADC Pulse Negative Counter 0 (ADCn_PCTNCT0)

Figure 36-35. ADC Pulse Negative Counter 0 (ADCn_PCTNCT0)

ADCn_PCTNCT0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	D[4]	D[3]	D[2]	D[1]	D[0]
Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 36-35. ADC Pulse Negative Counter 0 (ADCn_PCTNCT0) bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	D	ADC Pulse Negative Counter Register This register reflects the current counter value of the negative counter. These bits are read-only. Reload value is determined by ADCn_PCTNRL0.

ADC Pulse Positive Counter (ADCn_PCTPCT0)

Figure 36-36. ADC Pulse Positive Counter 0 (ADCn_PCTPCT0)

ADCn_PCTPCT0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 36-36. ADC Pulse Positive Counter 0 (ADCn_PCTPCT0) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	D	ADC Pulse Positive Counter Register This register reflects the current counter value of the positive counter. These bits are read-only. Reload value is determined by ADCn_PCTPRL0.

36.2.27 ADC Pulse Counter Zero Flag Registers (ADCn_PCZF32, ADCn_PCZF10)

These registers contain the zero flags which are set whenever ADCn_PCTPCT31~0 counters (positive counter) become zero i.e. the desired pulse length was detected. Each bit corresponds to the appropriate ADC analog input channels.

ADC Pulse Counter Zero Flag Register (ADCn_PCZF32)

Figure 36-37. ADC Pulse Counter Zero Flag Register (ADCn_PCZF32)

ADCn_PCZF32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPZF0	CTPZF1	CTPZF2	CTPZF3	CTPZF4	CTPZF5	CTPZF6	CTPZF7	CTPZF8	CTPZF9	CTPZF10	CTPZF11	CTPZF12	CTPZF13	CTPZF14	CTPZF15
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-37. ADC Pulse Counter Zero Flag Register (ADCn_PCZF32) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPZF0	Positive Counter Zero Flag for ADC channel 0 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of ADC channel 0 is set. This bit is cleared by the CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC0 bit.
[14]	CTPZF1	Positive Counter Zero Flag for ADC Channel 1 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 1 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC1 bit.
[13]	CTPZF2	Positive Counter Zero Flag for ADC Channel 2 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 2 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC2 bit.

Table 36-37. ADC Pulse Counter Zero Flag Register (ADCn_PCZF32) bits

Bit Position	Bit Field Name	Bit Description
[12]	CTPZF3	Positive Counter Zero Flag for ADC Channel 3 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 3 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC3 bit.
[11]	CTPZF4	Positive Counter Zero Flag for ADC Channel 4 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 4 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC4 bit.
[10]	CTPZF5	Positive Counter Zero Flag for ADC Channel 5 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 5 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC5 bit.
[9]	CTPZF6	Positive Counter Zero Flag for ADC Channel 6 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 6 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC6 bit.
[8]	CTPZF7	Positive Counter Zero Flag for ADC Channel 7 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 7 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC7 bit.

Table 36-37. ADC Pulse Counter Zero Flag Register (ADCn_PCZF32) bits

Bit Position	Bit Field Name	Bit Description
[7]	CTPZF8	Positive Counter Zero Flag for ADC Channel 8 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 8 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC8 bit.
[6]	CTPZF9	Positive Counter Zero Flag for ADC Channel 9 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 9 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC9 bit.
[5]	CTPZF10	Positive Counter Zero Flag for ADC Channel 10 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 10 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC10 bit.
[4]	CTPZF11	Positive Counter Zero Flag for ADC Channel 11 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 11 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC11 bit.
[3]	CTPZF12	Positive Counter Zero Flag for ADC Channel 12 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 12 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC12 bit.

Table 36-37. ADC Pulse Counter Zero Flag Register (ADCn_PCZF32) bits

Bit Position	Bit Field Name	Bit Description
[2]	CTPZF13	Positive Counter Zero Flag for ADC Channel 13 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 13 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC13 bit.
[1]	CTPZF14	Positive Counter Zero Flag for ADC Channel 14 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 14 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC14 bit.
[0]	CTPZF15	Positive Counter Zero Flag for ADC Channel 15 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 15 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC32:CTPZFC15 bit.

ADC Pulse Counter Zero Flag Register (ADCn_PCZF10)

Figure 36-38. ADC Pulse Counter Zero Flag Register (ADCn_PCZF10)

ADCn_PCZF10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPZF16	CTPZF17	CTPZF18	CTPZF19	CTPZF20	CTPZF21	CTPZF22	CTPZF23	CTPZF24	CTPZF25	CTPZF26	CTPZF27	CTPZF28	CTPZF29	CTPZF30	CTPZF31
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-38. ADC Pulse Counter Zero Flag Register (ADCn_PCZF10) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPZF16	Positive Counter Zero Flag for ADC Channel 16 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of ADC channel 16 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC16 bit.
[14]	CTPZF17	Positive Counter Zero Flag for ADC Channel 17 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 17 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC17 bit.
[13]	CTPZF18	Positive Counter Zero Flag for ADC Channel 18 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 18 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC18 bit.
[12]	CTPZF19	Positive Counter Zero Flag for ADC Channel 19 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 19 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC19 bit.
[11]	CTPZF20	Positive Counter Zero Flag for ADC Channel 20 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 20 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC20 bit.

Table 36-38. ADC Pulse Counter Zero Flag Register (ADCn_PCZF10) bits

Bit Position	Bit Field Name	Bit Description
[10]	CTPZF21	Positive Counter Zero Flag for ADC Channel 21 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 21 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC21 bit.
[9]	CTPZF22	Positive Counter Zero Flag for ADC Channel 22 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 22 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC22 bit.
[8]	CTPZF23	Positive Counter Zero Flag for ADC Channel 23 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 23 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC bit23.
[7]	CTPZF24	Positive Counter Zero Flag for ADC Channel 24 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 24 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC24 bit.
[6]	CTPZF25	Positive Counter Zero Flag for ADC Channel 25 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 25 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC25 bit.

Table 36-38. ADC Pulse Counter Zero Flag Register (ADCn_PCZF10) bits

Bit Position	Bit Field Name	Bit Description
[5]	CTPZF26	Positive Counter Zero Flag for ADC Channel 26 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 26 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC26 bit.
[4]	CTPZF27	Positive Counter Zero Flag for ADC Channel 27 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 27 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC27 bit.
[3]	CTPZF28	Positive Counter Zero Flag for ADC Channel 28 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 28 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC28 bit.
[2]	CTPZF29	Positive Counter Zero Flag for ADC Channel 29 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 29 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC29 bit.
[1]	CTPZF30	Positive Counter Zero Flag for ADC Channel 30 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 30 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC30 bit.

Table 36-38. ADC Pulse Counter Zero Flag Register (ADCn_PCZF10) bits

Bit Position	Bit Field Name	Bit Description
[0]	CTPZF31	Positive Counter Zero Flag for ADC Channel 31 This bit returns the status of the zero flag which is set when positive counter decrements to zero. The positive counter and negative counter are stopped as long as the zero flag of the appropriate channel 31 is set. This bit is cleared by CPU by writing '1' to the corresponding ADCn_PCZFC10:CTPZFC31 bit.

Note: All the bits in the registers ADCn_PCZF32 and ADCn_PCZF10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.2.28 ADC Pulse Counter Zero Flag Clear Registers (ADCn_PCZFC32, ADCn_PCZFC10)

These registers contain the bits to clear the zero flags in ADCn_PCZF32 and ADCn_PCZF10 registers. Each bit corresponds to the appropriate zero flag in ADCn_PCZF32 and ADCn_PCZF10 registers.

ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC32)

Figure 36-39. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC32)

ADCn_PCZFC32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPZFC0	CTPZFC1	CTPZFC2	CTPZFC3	CTPZFC4	CTPZFC5	CTPZFC6	CTPZFC7	CTPZFC8	CTPZFC9	CTPZFC10	CTPZF11	CTPZFC12	CTPZFC13	CTPZFC14	CTPZFC15
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-39. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC32)

Bit Position	Bit Field Name	Bit Description
[15]	CTPZFC0	Positive Counter Zero Flag Clear for ADC Channel 0 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF0 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[14]	CTPZFC1	Positive Counter Zero Flag Clear for ADC Channel 1 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF1 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[13]	CTPZFC2	Positive Counter Zero Flag Clear for ADC Channel 2 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF2 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.

Table 36-39. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC32)

Bit Position	Bit Field Name	Bit Description
[12]	CTPZFC3	Positive Counter Zero Flag Clear for ADC Channel 3 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF3 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[11]	CTPZFC4	Positive Counter Zero Flag Clear for ADC Channel 4 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF4 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[10]	CTPZFC5	Positive Counter Zero Flag Clear for ADC Channel 5 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF5 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[9]	CTPZFC6	Positive Counter Zero Flag Clear for ADC Channel 6 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF6 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[8]	CTPZFC7	Positive Counter Zero Flag Clear for ADC Channel 7 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF7 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[7]	CTPZFC8	Positive Counter Zero Flag Clear for ADC Channel 8 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF8 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[6]	CTPZFC9	Positive Counter Zero Flag Clear for ADC Channel 9 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF9 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.

Table 36-39. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC32)

Bit Position	Bit Field Name	Bit Description
[5]	CTPZFC10	Positive Counter Zero Flag Clear for ADC Channel 10 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF10 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[4]	CTPZF11	Positive Counter Zero Flag Clear for ADC Channel 11 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF11 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[3]	CTPZFC12	Positive Counter Zero Flag Clear for ADC Channel 12 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF12 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[2]	CTPZFC13	Positive Counter Zero Flag Clear for ADC Channel 13 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF13 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[1]	CTPZFC14	Positive Counter Zero Flag Clear for ADC Channel 14 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF14 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[0]	CTPZFC15	Positive Counter Zero Flag Clear for ADC Channel 15 '0': No effect '1': Clears the corresponding ADCn_PCZF32:CTPZF15 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.

ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC10)

Figure 36-40. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC10)

ADCn_PCZFC10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPZFC16	CTPZFC17	CTPZFC18	CTPZFC19	CTPZFC20	CTPZFC21	CTPZFC22	CTPZFC23	CTPZFC24	CTPZFC25	CTPZFC26	CTPZFC27	CTPZFC28	CTPZFC29	CTPZFC30	CTPZFC31
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-40. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC10) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPZFC16	Positive Counter Zero Flag Clear for ADC Channel 16 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF16 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[14]	CTPZFC17	Positive Counter Zero Flag Clear for ADC Channel 17 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF17 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[13]	CTPZFC18	Positive Counter Zero Flag Clear for ADC Channel 18 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF18 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[12]	CTPZFC19	Positive Counter Zero Flag Clear for ADC Channel 19 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF19 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.

Table 36-40. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC10) bits

Bit Position	Bit Field Name	Bit Description
[11]	CTPZFC20	Positive Counter Zero Flag Clear for ADC Channel 20 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF20 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[10]	CTPZFC21	Positive Counter Zero Flag Clear for ADC Channel 21 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF21 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[9]	CTPZFC22	Positive Counter Zero Flag Clear for ADC Channel 22 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF22 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[8]	CTPZFC23	Positive Counter Zero Flag Clear for ADC Channel 23 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF23 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[7]	CTPZFC24	Positive Counter Zero Flag Clear for ADC Channel 24 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF24 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[6]	CTPZFC25	Positive Counter Zero Flag Clear for ADC Channel 25 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF25 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[5]	CTPZFC26	Positive Counter Zero Flag Clear for ADC Channel 26 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF26 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.

Table 36-40. ADC Pulse Counter Zero Flag Clear Register (ADCn_PCZFC10) bits

Bit Position	Bit Field Name	Bit Description
[4]	CTPZFC27	Positive Counter Zero Flag Clear for ADC Channel 27 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF27 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[3]	CTPZFC28	Positive Counter Zero Flag Clear for ADC Channel 28 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF28 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[2]	CTPZFC29	Positive Counter Zero Flag Clear for ADC Channel 29 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF29 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[1]	CTPZFC30	Positive Counter Zero Flag Clear for ADC Channel 30 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF30 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.
[0]	CTPZFC31	Positive Counter Zero Flag Clear for ADC Channel 31 '0': No effect '1': Clears the corresponding ADCn_PCZF10:CTPZF31 bit and reloads the positive and negative counters with their reload values Reading always returns '0'.

Note: All the bits in the registers ADCn_PCZFC32 and ADCn_PCZFC10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.2.29 ADC Pulse Counter Interrupt Enable Registers (ADCn_PCIE32, ADCn_PCIE10)

These registers contain the bits to enable the generation of interrupt from the corresponding ADC pulse detection function module. If the appropriate ADC pulse counter zero flag is set and the interrupt is enabled, ADPINT interrupt is asserted.

ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32)

Figure 36-41. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32)

ADCn_PCIE32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPIE0	CTPIE1	CTPIE2	CTPIE3	CTPIE4	CTPIE5	CTPIEC6	CTPIE7	CTPIE8	CTPIE9	CTPIE10	CTPIE11	CTPIE12	CTPIE13	CTPIE14	CTPIE15
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-41. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPIE0	ADC Pulse Counter Interrupt Enable for ADC Channel 0 '0':The pulse counter interrupt for ADC Channel 0 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 0 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES0. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC0.
[14]	CTPIE1	ADC Pulse Counter Interrupt Enable for ADC Channel 1 '0':The pulse counter interrupt for ADC Channel 1 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 1 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES1. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC1.
[13]	CTPIE2	ADC Pulse Counter Interrupt Enable for ADC Channel 2 '0':The pulse counter interrupt for ADC Channel 2 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 2 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES2. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC2.

Table 36-41. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32) bits

Bit Position	Bit Field Name	Bit Description
[12]	CTPIE3	ADC Pulse Counter Interrupt Enable for ADC Channel 3 '0':The pulse counter interrupt for ADC Channel 3 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 3 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES3. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC3.
[11]	CTPIE4	ADC Pulse Counter Interrupt Enable for ADC Channel 4 '0':The pulse counter interrupt for ADC Channel 4 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 4 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES4. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC4.
[10]	CTPIE5	ADC Pulse Counter Interrupt Enable for ADC Channel 5 '0':The pulse counter interrupt for ADC Channel 5 is disabled '1':Enables the generation of he pulse counter interrupt for ADC Channel 5 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES5. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC5.
[9]	CTPIEC6	ADC Pulse Counter Interrupt Enable for ADC Channel 6 '0':The pulse counter interrupt for ADC Channel 6 is disabled '1':Enables the generation of he pulse counter interrupt for ADC Channel 6 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES6. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC6.
[8]	CTPIE7	ADC Pulse Counter Interrupt Enable for ADC Channel 7 '0':The pulse counter interrupt for ADC Channel 7 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 7 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES7. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC7.

Table 36-41. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32) bits

Bit Position	Bit Field Name	Bit Description
[7]	CTPIE8	ADC Pulse Counter Interrupt Enable for ADC Channel 8 '0':The pulse counter interrupt for ADC Channel 8 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 8 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES8. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC8.
[6]	CTPIE9	ADC Pulse Counter Interrupt Enable for ADC Channel 9 '0':The pulse counter interrupt for ADC Channel 9 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 9 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES9. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC9.
[5]	CTPIE10	ADC Pulse Counter Interrupt Enable for ADC Channel 10 '0':The pulse counter interrupt for ADC Channel 10 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 10 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES10. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC10.
[4]	CTPIE11	ADC Pulse Counter Interrupt Enable for ADC Channel 11 '0':The pulse counter interrupt for ADC Channel 11 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 11 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES11. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC11.
[3]	CTPIE12	ADC Pulse Counter Interrupt Enable for ADC Channel 12 '0':The pulse counter interrupt for ADC Channel 12 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 12 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES12. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC12.

Table 36-41. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32) bits

Bit Position	Bit Field Name	Bit Description
[2]	CTPIE13	ADC Pulse Counter Interrupt Enable for ADC Channel 13 '0':The pulse counter interrupt for ADC Channel 13 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 13 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES13. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC13.
[1]	CTPIE14	ADC Pulse Counter Interrupt Enable for ADC Channel 14 '0':The pulse counter interrupt for ADC Channel 14 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 14 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES14. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC14.
[0]	CTPIE15	ADC Pulse Counter Interrupt Enable for ADC Channel 15 '0':The pulse counter interrupt for ADC Channel 15 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 15 This bit can also be set by writing '1' to ADCn_PCIES32:CTPIES15. This bit can also be cleared by writing '1' to ADCn_PCIEC32:CTPIEC15.

ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE10)

Figure 36-42. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE10)

ADCn_PCIE10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPIE16	CTPIE17	CTPIE18	CTPIE19	CTPIE20	CTPIE21	CTPIE22	CTPIE23	CTPIE24	CTPIE25	CTPIE26	CTPIE27	CTPIE28	CTPIE29	CTPIE30	CTPIE31
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-42. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE10) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPIE16	ADC Pulse Counter Interrupt Enable for ADC Channel 16 '0':The pulse counter interrupt for ADC Channel 16 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 16 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES16. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC16.
[14]	CTPIE17	ADC Pulse Counter Interrupt Enable for ADC Channel 17 '0':The pulse counter interrupt for ADC Channel 17 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 17 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES17. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC17.
[13]	CTPIE18	ADC Pulse Counter Interrupt Enable for ADC Channel 18 '0':The pulse counter interrupt for ADC Channel 18 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 18 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES18. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC18.
[12]	CTPIE19	ADC Pulse Counter Interrupt Enable 19 '0':The pulse counter interrupt for ADC Channel 19 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 19 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES19. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC19.
[11]	CTPIE20	ADC Pulse Counter Interrupt Enable for ADC Channel 20 '0':The pulse counter interrupt for ADC Channel 20 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 20 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES20. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC20.

Table 36-42. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE10) bits

Bit Position	Bit Field Name	Bit Description
[10]	CTPIE21	ADC Pulse Counter Interrupt Enable for ADC Channel 21 '0':The pulse counter interrupt for ADC Channel 21 is disabled '1':Enables the generation the pulse counter interrupt for ADC Channel 21 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES21. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC21.
[9]	CTPIE22	ADC Pulse Counter Interrupt Enable for ADC Channel 22 '0':The pulse counter interrupt for ADC Channel 22 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 22 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES22. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC22.
[8]	CTPIE23	ADC Pulse Counter Interrupt Enable for ADC Channel 23 '0':The pulse counter interrupt for ADC Channel 23 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 23 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES23. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC23.
[7]	CTPIE24	ADC Pulse Counter Interrupt Enable for ADC Channel 24 '0':The pulse counter interrupt for ADC Channel 24 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 24 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES24. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC24.
[6]	CTPIE25	ADC Pulse Counter Interrupt Enable for ADC Channel 25 '0':The pulse counter interrupt for ADC Channel 25 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 25 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES25. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC25.

Table 36-42. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE10) bits

Bit Position	Bit Field Name	Bit Description
[5]	CTPIE26	ADC Pulse Counter Interrupt Enable for ADC Channel 26 '0':The pulse counter interrupt for ADC Channel 26 is disabled '1':Enables the generation the pulse counter interrupt for ADC Channel 26 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES26. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC26.
[4]	CTPIE27	ADC Pulse Counter Interrupt Enable for ADC Channel 27 '0':The pulse counter interrupt for ADC Channel 27 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 27 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES27. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC27.
[3]	CTPIE28	ADC Pulse Counter Interrupt Enable for ADC Channel 28 '0':The pulse counter interrupt for ADC Channel 28 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 28 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES28. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC28.
[2]	CTPIE29	ADC Pulse Counter Interrupt Enable for ADC Channel 29 '0':The pulse counter interrupt for ADC Channel 29 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 29 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES29. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC29.
[1]	CTPIE30	ADC Pulse Counter Interrupt Enable for ADC Channel 30 '0':The pulse counter interrupt for ADC Channel 30 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 30 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES30. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC30.

Table 36-42. ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE10) bits

Bit Position	Bit Field Name	Bit Description
[0]	CTPIE31	ADC Pulse Counter Interrupt Enable for ADC Channel 31 '0':The pulse counter interrupt for ADC Channel 31 is disabled '1':Enables the generation of the pulse counter interrupt for ADC Channel 31 This bit can also be set by writing '1' to ADCn_PCIES10:CTPIES31. This bit can also be cleared by writing '1' to ADCn_PCIEC10:CTPIEC31.

Note: All the bits in the registers ADCn_PCIE32 and ADCn_PCIE10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.2.30 ADC Pulse Counter Interrupt Enable Set Registers (ADCn_PCIES32, ADCn_PCIES10)

The ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES32, ADCn_PCIES10) contains bits to set the bits in ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32, ADCn_PCIE10).

ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES32)

Figure 36-43. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES32)

ADCn_PCIES32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPIES0	CTPIES1	CTPIES2	CTPIES3	CTPIES4	CTPIES5	CTPIES6	CTPIES7	CTPIES8	CTPIES9	CTPIES10	CTPIES11	CTPIES12	CTPIES13	CTPIES14	CTPIES15
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-43. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES32) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPIES0	ADC Pulse Counter Interrupt Enable Set for ADC Channel 0 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE Reading this bit always returns '0'.
[14]	CTPIES1	ADC Pulse Counter Interrupt Enable Set for ADC Channel 1 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE1 Reading this bit always returns '0'.
[13]	CTPIES2	ADC Pulse Counter Interrupt Enable Set for ADC Channel 2 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE2 Reading this bit always returns '0'.
[12]	CTPIES3	ADC Pulse Counter Interrupt Enable Set for ADC Channel 3 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE3 Reading this bit always returns '0'.

Table 36-43. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES32) bits

Bit Position	Bit Field Name	Bit Description
[11]	CTPIES4	ADC Pulse Counter Interrupt Enable Set for ADC Channel 4 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE4 Reading this bit always returns '0'.
[10]	CTPIES5	ADC Pulse Counter Interrupt Enable Set for ADC Channel 5 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE5 Reading this bit always returns '0'.
[9]	CTPIES6	ADC Pulse Counter Interrupt Enable Set for ADC Channel 6 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE6 Reading this bit always returns '0'.
[8]	CTPIES7	ADC Pulse Counter Interrupt Enable Setfor ADC Channel 7 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE7 Reading this bit always returns '0'.
[7]	CTPIES8	ADC Pulse Counter Interrupt Enable Set for ADC Channel 8 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE8 Reading this bit always returns '0'.
[6]	CTPIES9	ADC Pulse Counter Interrupt Enable Set for ADC Channel 9 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE9 Reading this bit always returns '0'.
[5]	CTPIES10	ADC Pulse Counter Interrupt Enable Set for ADC Channel 10 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE10 Reading this bit always returns '0'.
[4]	CTPIES11	ADC Pulse Counter Interrupt Enable Set for ADC Channel 11 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE11 Reading this bit always returns '0'.
[3]	CTPIES12	ADC Pulse Counter Interrupt Enable Set for ADC Channel 12 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE12 Reading this bit always returns '0'.

Table 36-43. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES32) bits

Bit Position	Bit Field Name	Bit Description
[2]	CTPIES13	ADC Pulse Counter Interrupt Enable Set for ADC Channel 13 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE13 Reading this bit always returns '0'.
[1]	CTPIES14	ADC Pulse Counter Interrupt Enable Set for ADC Channel 14 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE14 Reading this bit always returns '0'.
[0]	CTPIES15	ADC Pulse Counter Interrupt Enable Set for ADC Channel 15 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE32:CTPIE15 Reading this bit always returns '0'.

ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES10)

Figure 36-44. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES10)

ADCn_PCIES10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPIES16	CTPIES17	CTPIES18	CTPIES19	CTPIES20	CTPIES21	CTPIES22	CTPIES23	CTPIES24	CTPIES25	CTPIES26	CTPIES27	CTPIES28	CTPIES29	CTPIES30	CTPIES31
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-44. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES10) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPIES16	ADC Pulse Counter Interrupt Enable Set for ADC Channel 16 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE16 Reading this bit always returns '0'.
[14]	CTPIES17	ADC Pulse Counter Interrupt Enable Set for ADC Channel 17 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE17 Reading this bit always returns '0'.

Table 36-44. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES10) bits

Bit Position	Bit Field Name	Bit Description
[13]	CTPIES18	ADC Pulse Counter Interrupt Enable Set for ADC Channel 18 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE18 Reading this bit always returns '0'.
[12]	CTPIES19	ADC Pulse Counter Interrupt Enable Set for ADC Channel 19 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE19 Reading this bit always returns '0'.
[11]	CTPIES20	ADC Pulse Counter Interrupt Enable Set for ADC Channel 20 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE20 Reading this bit always returns '0'.
[10]	CTPIES21	ADC Pulse Counter Interrupt Enable Set for ADC Channel 21 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE21 Reading this bit always returns '0'.
[9]	CTPIES22	ADC Pulse Counter Interrupt Enable Set for ADC Channel 22 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE22 Reading this bit always returns '0'.
[8]	CTPIES23	ADC Pulse Counter Interrupt Enable Set for ADC Channel 23 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE23 Reading this bit always returns '0'.
[7]	CTPIES24	ADC Pulse Counter Interrupt Enable Set for ADC Channel 24 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE24 Reading this bit always returns '0'.
[6]	CTPIES25	ADC Pulse Counter Interrupt Enable Set for ADC Channel 25 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE25 Reading this bit always returns '0'.
[5]	CTPIES26	ADC Pulse Counter Interrupt Enable Set for ADC Channel 26 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE26 Reading this bit always returns '0'.

Table 36-44. ADC Pulse Counter Interrupt Enable Set Register (ADCn_PCIES10) bits

Bit Position	Bit Field Name	Bit Description
[4]	CTPIES27	ADC Pulse Counter Interrupt Enable Set for ADC Channel 27 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE27 Reading this bit always returns '0'.
[3]	CTPIES28	ADC Pulse Counter Interrupt Enable Set for ADC Channel 28 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE28 Reading this bit always returns '0'.
[2]	CTPIES29	ADC Pulse Counter Interrupt Enable Set for ADC Channel 29 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE29 Reading this bit always returns '0'.
[1]	CTPIES30	ADC Pulse Counter Interrupt Enable Set for ADC Channel 30 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE30 Reading this bit always returns '0'.
[0]	CTPIES31	ADC Pulse Counter Interrupt Enable Set for ADC Channel 31 '0': No effect '1': Sets the corresponding bit in ADCn_PCIE10:CTPIE31 Reading this bit always returns '0'.

Note: All the bits in the registers ADCn_PCIES32 and ADCn_PCIES10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.2.31 ADC Pulse Counter Interrupt Enable Clear Registers (ADCn_PCIEC32, ADCn_PCIEC10)

The ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC32, ADCn_PCIEC10) contains bits to clear the bits in ADC Pulse Counter Interrupt Enable Register (ADCn_PCIE32, ADCn_PCIE10).

ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC32)

Figure 36-45. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC32)

ADCn_PCIEC32															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPIEC0	CTPIEC1	CTPIEC2	CTPIEC3	CTPIEC4	CTPIEC5	CTPIE6	CTPIEC7	CTPIEC8	CTPIEC9	CTPIEC10	CTPIEC11	CTPIEC12	CTPIEC13	CTPIEC14	CTPIEC15
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-45. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC32) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPIEC0	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 0 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE0 Reading this bit always returns '0'.
[14]	CTPIEC1	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 1 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE1 Reading this bit always returns '0'.
[13]	CTPIEC2	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 2 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE2 Reading this bit always returns '0'.
[12]	CTPIEC3	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 3 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE3 Reading this bit always returns '0'.

Table 36-45. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC32) bits

Bit Position	Bit Field Name	Bit Description
[11]	CTPIEC4	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 4 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE4 Reading this bit always returns '0'.
[10]	CTPIEC5	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 5 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE5 Reading this bit always returns '0'.
[9]	CTPIE6	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 6 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE6 Reading this bit always returns '0'.
[8]	CTPIEC7	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 7 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE7 Reading this bit always returns '0'.
[7]	CTPIEC8	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 8 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE8 Reading this bit always returns '0'.
[6]	CTPIEC9	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 9 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE9 Reading this bit always returns '0'.
[5]	CTPIEC10	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 10 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE10 Reading this bit always returns '0'.
[4]	CTPIEC11	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 11 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE11 Reading this bit always returns '0'.
[3]	CTPIEC12	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 12 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE12 Reading this bit always returns '0'.

Table 36-45. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC32) bits

Bit Position	Bit Field Name	Bit Description
[2]	CTPIEC13	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 13 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE13 Reading this bit always returns '0'.
[1]	CTPIEC14	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 14 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE14 Reading this bit always returns '0'.
[0]	CTPIEC15	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 15 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE32:CTPIE15 Reading this bit always returns '0'.

ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC10)

Figure 36-46. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC10)

ADCn_PCIEC10															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
CTPIEC16	CTPIEC17	CTPIEC18	CTPIEC19	CTPIEC20	CTPIEC21	CTPIEC22	CTPIEC23	CTPIEC24	CTPIEC25	CTPIEC26	CTPIEC27	CTPIEC28	CTPIEC29	CTPIEC30	CTPIEC31
Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36-46. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC10) bits

Bit Position	Bit Field Name	Bit Description
[15]	CTPIEC16	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 16 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE16 Reading this bit always returns '0'.
[14]	CTPIEC17	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 17 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE17 Reading this bit always returns '0'.

Table 36-46. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC10) bits

Bit Position	Bit Field Name	Bit Description
[13]	CTPIEC18	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 18 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE18 Reading this bit always returns '0'.
[12]	CTPIEC19	ADC Pulse Counter Interrupt Enable Clear 19 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE19 Reading this bit always returns '0'.
[11]	CTPIEC20	ADC Pulse Counter Interrupt Enable Clear 20 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE20 Reading this bit always returns '0'.
[10]	CTPIEC21	ADC Pulse Counter Interrupt Enable Clear 21 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE21 Reading this bit always returns '0'.
[9]	CTPIEC22	ADC Pulse Counter Interrupt Enable Clear 22 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE22 Reading this bit always returns '0'.
[8]	CTPIEC23	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 23 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE23 Reading this bit always returns '0'.
[7]	CTPIEC24	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 24 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE24 Reading this bit always returns '0'.
[6]	CTPIEC25	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 25 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE25 Reading this bit always returns '0'.
[5]	CTPIEC26	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 26 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE26 Reading this bit always returns '0'.

Table 36-46. ADC Pulse Counter Interrupt Enable Clear Register (ADCn_PCIEC10) bits

Bit Position	Bit Field Name	Bit Description
[4]	CTPIEC27	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 27 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE27 Reading this bit always returns '0'.
[3]	CTPIEC28	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 28 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE28 Reading this bit always returns '0'.
[2]	CTPIEC29	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 29 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE29 Reading this bit always returns '0'.
[1]	CTPIEC30	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 30 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE30 Reading this bit always returns '0'.
[0]	CTPIEC31	ADC Pulse Counter Interrupt Enable Clear for ADC Channel 31 '0': No effect '1': Clears the corresponding bit in ADCn_PCIE10:CTPIE31 Reading this bit always returns '0'.

Note: All the bits in the registers ADCn_PCIEC32 and ADCn_PCIEC10 are reversed. For more details on reversal of bits refer to Access of registers by ARM CLZ instruction in 36.3 Operation of A/D Converter.

36.3 Operation of A/D Converter

The A/D Converter operates using the successive approximation method with 10-bit or bit resolution. There is one 16-bit register provided to store conversion results (ADCn_CR), which is updated each time conversion completes. Additionally, there is one ADC Channel Data Register per channel (ADCn_CD31~0), which is updated each time the assigned channel is converted. The Channel Data Registers especially improve the continuous conversion mode.

The range comparator compares the converted values with the configured values in the threshold registers. It then accordingly generates an interrupt for 'inside range' or 'outside range' depending on the configuration of ADCn_RCOIRS32, ADCn_RCOIRS10 registers. There are four range comparators present, where any of the range comparators can be configured for any of the available ADC analog input channels. The ADC pulse detection function detects events of desired length and also filters parasitic inverted events. Each ADC channel has a dedicated pulse detection module. The following section describes the operation of the A/D Converter.

Single mode

In single conversion mode, the analog input channels which are enabled by ADCn_ER32, ADCn_ER10:ADE bits are selected and converted in order. Upon assertion of the start signal selected by the ADCn_CS1:STS[1:0] bits, the conversion starts from the start channel which is configured by ADCn_SCH:ANS bits and stops when the analog input conversion of the end channel configured by ADCn_ECH:ANE is finished. The analog input signals which are disabled by ADCn_ER32 and ADCn_ER10 are skipped in the range set between ADCn_SCH:ANS and ADCn_ECH:ANE. If the start channel and end channel are the same (ADCn_SCH:ANS = ADCn_ECH:ANE), only a single channel conversion is performed. For the ADC to start conversion, the channel selected by ADCn_SCH:ANS[4:0] bits (start channel) must be enabled by the corresponding ADCn_ER32 and ADCn_ER10:ADE bit.

Examples:

ADCn_ER32:ADE[0:3] = '1111'

ADCn_SCH:ANS = '00000', ADCn_ECH:ANE = '00011'

Start -> AN0 -> AN1 -> AN2 -> AN3 -> end

ADCn_SCH:ANS = '00010', ADCn_ECH:ANE = '00010'

Start -> AN2 -> end ADCn_ER32:ADE[0:3] = '1011'

ADCn_SCH:ANS = '00000', ADCn_ECH:ANE = '00011'

Start -> AN0 ->AN2 -> AN3 -> end

ADCn_ER32:ADE[2] = '1'

ADCn_SCH:ANS = '00010', ADCn_ECH:ANE = '00010'

Start ->AN2 ->end

Note: In single mode, for conversion of one scan sequence a single trigger is required which is determined by the ADCn_CS1:STS[1:0] bits. This mode enables synchronization of the conversion start signal.

Continuous mode

In continuous mode the analog input channels which are enabled by ADCn_ER32 and ADCn_ER10 are selected and converted in order. Upon assertion of the start signal selected by the ADCn_CS1:STS[1:0] bits, the conversion starts from the start channel which is configured by ADCn_SCH:ANS and stops when the conversion of the end channel configured by ADCn_ECH:ANE is finished. After this the converter returns to the ADCn_SCH:ANS channel and repeats the process continuously. When the start and end channels are the same (ADCn_SCH:ANS = ADCn_ECH:ANE), conversion is performed continuously for that channel. For the ADC to start conversion, the channel selected by ADCn_SCH:ANS[4:0] bits (start channel) must be enabled by the corresponding ADCn_ER32 and ADCn_ER10:ADE bit.

Examples: ADCn_ER32:ADE[0:3] = '1111'

ADCn_SCH:ANS = '00000', ADCn_ECH:ANE = '00011'

Start -> AN0 -> AN1 -> AN2 -> AN3 -> AN0... -> repeat

ADCn_SCH:ANS = '00010', ADCn_ECH:ANE = '00010'

Start -> AN2 -> AN2 -> AN2... -> repeat

ADCn_ER32:ADE[0:3] = '1011'

ADCn_SCH:ANS = ’00000', ADCn_ECH:ANE = '00011'

Start -> AN0 -> AN2 ->AN3 ->AN0... ->repeat

ADCn_ER32:ADE[2] = '1'

ADCn_SCH:ANS = '00010', ADCn_ECH:ANE = '00010'

Start -> AN2 -> AN2 -> AN2... -> repeat

In continuous mode, conversion is repeated until '1' is written to the ADCn_CSC1:BUSYC bit which clears ADCn_CS1:BUSY (clearing the BUSY bit forcibly stops the conversion operation).

Notes:

1. Forcibly terminating operation halts the current conversion during mid-conversion (if operation is forcibly terminated, the value in the conversion register is the result of the most recently completed conversion).
2. In continuous mode, for conversion of one scan sequence a single trigger is required which is determined by the ADCn_CS1:STS[1:0] bits. This mode enables synchronization of the conversion start signal.

Stop mode

In stop mode, upon assertion of the start signal selected by the ADCn_CS1:STS[1:0] bits, the analog input channels selected by the ADCn_SCH:ANS bits and ADCn_ECH:ANE bits are converted in order, but conversion operation pauses after each channel. The pause is released by applying another start signal.

After the conversion on the end channel determined by the ADCn_ECH:ANE bits, the converter returns to the ADCn_SCH:ANS channel and repeats the conversion process continuously. When the start and end channels are the same (ADCn_SCH:ANS = ADCn_ECH:ANE), only a single channel conversion is performed.

In stop mode, each conversion is triggered by the trigger source determined by the ADCn_CS1:STS[1:0] bits. This mode enables synchronization of the conversion start signal.

Examples:

ADCn_SCH:ANS = '00000', ADCn_ECH:ANE = '00011'

Start -> AN0 -> stop -> start -> AN1 -> stop -> start -> AN2 -> stop -> start -> AN3 -> stop -> start - > AN0... -> repeat

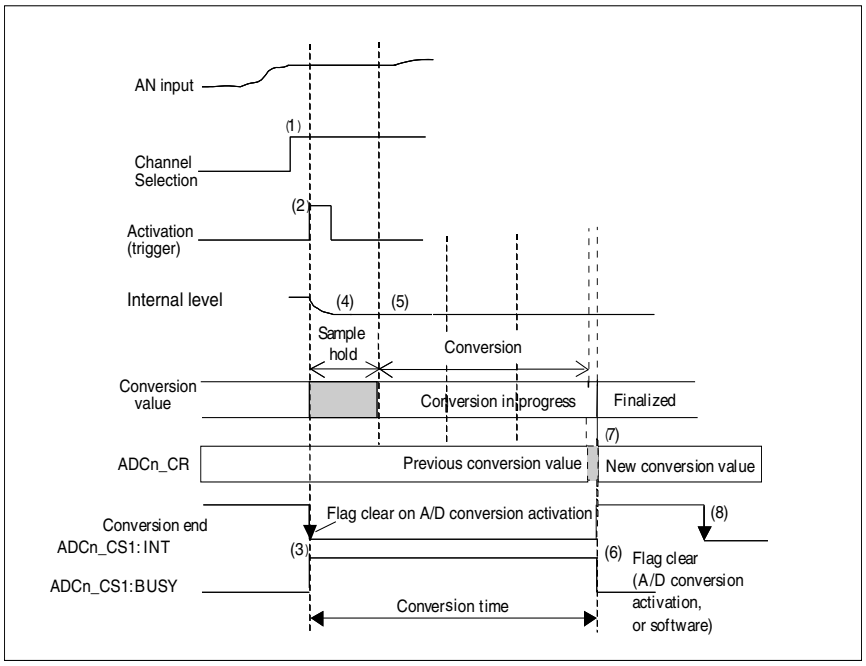
ADCn_SCH:ANS = '00010', ADCn_ECH:ANE = '00010'

Start -> AN2 -> stop -> start -> AN2 -> stop -> start -> AN2... -> repeat

The channels that are enabled by ADCn_ER32, ADCn_ER10:ADE bits are only converted in stop mode. The conversion is never started if the start channel (set by ADCn_SCH:ANS) is set to '0'.

Single-shot conversion

Figure 36-47. of A/D Converter in single-shot conversion mode

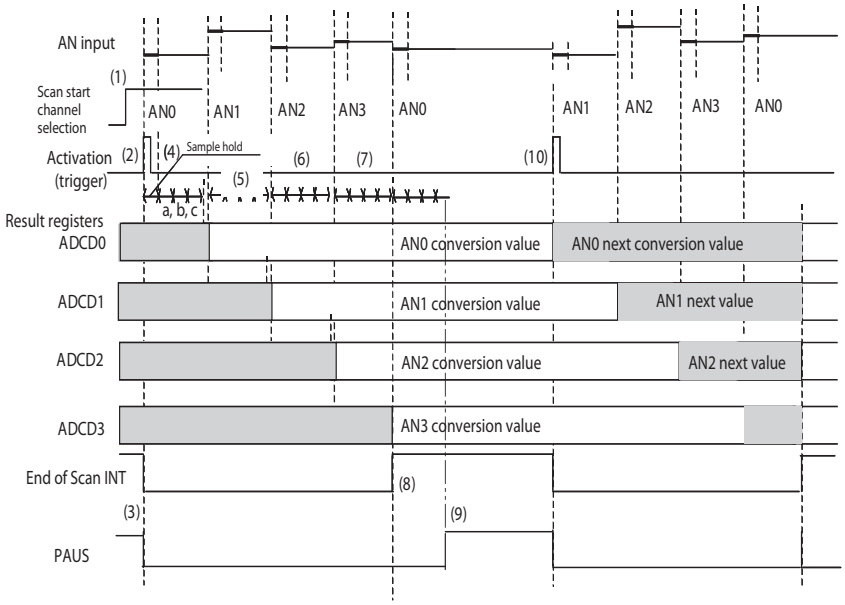


1. Channel selection.
2. A/D conversion activation (trigger input: software trigger/reload timer/external trigger).
3. ADCn_CS1:INT flag clear, ADCn_CS1:BUSY flag set.
4. Sample hold.
5. Conversion.
6. Conversion end, ADCn_CS1:INT flag set, ADCn_CS1:BUSY flag clear.
7. Store result in ADC Common Data Register (ADCn_CR).
8. Software-based ADCn_CS1:INT flag clear.

Note: In Figure 36-47 the ADCn_CS1:INT interrupt flag at the beginning (point 3) is shown to be cleared. This clearing is done by CPU after reading the previous converted value from the Common Data Register ADCn_CR by writing '1' to ADCn_CSC1:INTC.

Scan conversion

Figure 36-48. Operation of A/D Converter in scan conversion mode



1. Activation channel selection.
2. A/D activation (trigger: software trigger/reload timer/external trigger).
3. ADCn_CS1:INT flag clear, ADCn_CS1:PAUS flag clear.
4. AN0 conversion.
   a. Sample hold, conversion.
   b. Conversion end.
   c. Buffers the conversion value.
5. AN1 conversion.
6. AN2 conversion.
7. AN3 conversion.
8. ADCn_CS3:INT2 (end of scan) flag is set, AN0 conversion starts.
9. It is assumed that the data protection feature is enabled (ADCn_CS2:DPDIS = '0'). Since ADCn_CS3:INT2 has not been cleared yet,
10. ADC protects the value in ADCn_CD0 (data register of AN0) against overwriting, pauses the conversion operation and sets ADCn_CS1:PAUS bit. ADCn_CS3:INT2 flag is cleared by software, the ADC stores the result of AN0 in ADCn_CD0 and continues sampling AN1. The ADCn_CS1:PAUS bit is made low by the software to indicate a new occurance of pause condition to the user when it occurs.

Access of registers by Arm CLZ instruction

The range comparator result flags are organized ’per ADC channel’. There are 32 range comparator over threshold flags and 32 interrupt flags. In case of a range comparator interrupt, all interrupt flags can be read out by 32-bit read operation and analyzed using the Arm CLZ instruction. The Arm CLZ instruction returns the number of leading zeros starting from the left (MSB) i.e. it returns the next ADC channel which signaled an interrupt. Since Arm CLZ instruction works from MSB to LSB (from left to right), ADC channel 0 is located on the MSB of the registers and ADC channel 31 is on the LSB.

Due to this reason all the bits in the registers for 36.2.1 A/D Input Enable Registers (ADCn_ER32, ADCn_ER10), 36.2.21 Inverted Range Selection Registers (ADCn_RCOIRS32, ADCn_RCOIRS10), 36.2.22 Range Comparator Interrupt Flags (ADCn_RCOINT32, ADCn_RCOINT10), 36.2.23 Range Comparator Over Threshold Flags (ADCn_RCOOF32, ADCn_R-

COOF10), 36.2.24 Range Comparator Interrupt Clear Registers (ADCn_RCOINTC32, ADCn_RCOINTC10), 36.2.27 ADC Pulse Counter Zero Flag Registers (ADCn_PCZF32, ADCn_PCZF10), 36.2.28 ADC Pulse Counter Zero Flag Clear Registers (ADCn_PCZFC32, ADCn_PCZFC10), 36.2.29 ADC Pulse Counter Interrupt Enable Registers (ADCn_PCIE32, ADCn_PCIE10), 36.2.30 ADC Pulse Counter Interrupt Enable Set Registers (ADCn_PCIES32, ADCn_PCIES10), and 36.2.31 ADC Pulse Counter Interrupt Enable Clear Registers (ADCn_PCIEC32, ADCn_PCIEC10) are reversed.

Protection of the ADC Data Registers

There are 33 ADC data registers, one register per channel and one common register. The registers are written by hardware at the end of conversion of the attached channel. ADCn_CD0 is attached to channel 0, ADCn_CD31 is attached to channel 31.

The CPU can read the channel data registers any time.

If a conversion is finished and the data of the previous conversion has not been read out, previous data would be overwritten. To avoid this problem, the next conversion data is not stored in the data registers before the appropriate interrupt flag (ADCn_CS1:INT or ADCn_CS3:INT2) is cleared. A/D conversion halts during this time and the ADCn_CS1:PAUS flag is set.

The register protection function depends on the conversion mode and the setting of ADCn_CS3:INTE2, ADCn_CS1:INTE, ADCn_MAR:DRQEN, and also on ADCn_CS2:DPDIS bits. If ADCn_CS2:DPDIS = '1' then conversion is continued and former conversion data may be overwritten. Else if ADCn_CS2:DPDIS = '0' the former conversion data is not overwritten before the appropriate interrupt flag (ADCn_CS1:INT or ADCn_CS3:INT2) is cleared. The following table gives the protection scheme.

Table 36-47. Protection of the ADC Channel Data Register

Mode	ADCn_CS3: INTE2	ADCn_CS1: INTE	ADCn_MAR: DRQEN	Function
Continuous	'0'	'0'	'0'	ADCn_CS1:PAUS is not set because no interrupt request is generated (although the flags are set) and no DMA transfer is requested.
	'0'	'1'	'0'	ADCn_CS1:PAUS is set after each channel conversion when the interrupt request ADCn_CS1:INT of the preceding conversion has not been cleared. To resume conversion, the ADCn_CS1:INT flag must be cleared by writing '1' to ADCn_CSC1:INTC.
	'0'	X	'1'	ADCn_CS1:PAUS is set after each channel conversion when the interrupt request ADCn_CS1:INT of the preceding conversion has not been cleared. To resume conversion, the ADCn_CS1:INT flag must be cleared by writing '1' to ADCn_CSC1:INTC or by reading the ADCn_CR register (except by the debug master, DAP).
	'1'	'0'	'0'	ADCn_CS1:PAUS is set after the conversion of the start channel (defined by ADCn_SCH register) when the interrupt request ADCn_CS3:INT2 of the preceding scan has not been cleared. To resume conversion, the ADCn_CS3:INT2 flag must be cleared by writing '1' to ADCn_CSC3:INT2C.
	'1'	'1'	'0'	ADCn_CS1:PAUS is set after each channel conversion when the interrupt request ADCn_CS1:INT of the preceding conversion or the interrupt request ADCn_CS3:INT2 of the preceding scan has not been cleared. To resume conversion, the ADCn_CS1:INT and ADCn_CS3:INT2 flags must be cleared by writing '1' to ADCn_CSC1:INTC and ADCn_CSC3:INT2C respectively.
	'1'	X	'1'	ADCn_CS1:PAUS is set after each channel conversion when the interrupt request ADCn_CS1:INT of the preceding conversion or the interrupt request ADCn_CS3:INT2 of the preceding scan has not been cleared. To resume conversion, the ADCn_CS1:INT and ADCn_CS3:INT2 flags must be cleared. ADCn_CS1:INT is cleared by writing '1' to ADCn_CSC1:INTC or by reading the ADCn_CR register (except by the debug master, DAP). ADCn_CS3:INT2 flag is cleared by writing '1' to ADCn_CSC3:INT2C.

Table 36-47. Protection of the ADC Channel Data Register

Mode	ADCn_ CS3: INTE2	ADCn_ CS1: INTE	ADCn_ MAR: DRQEN	Function
Others	X	'0'	'0'	ADCn_CS1:PAUS is not set because no interrupt request is generated (although the flags are set) and no DMA transfer is requested.
	X	'1'	'0'	ADCn_CS1:PAUS is set after each channel conversion when the interrupt request ADCn_CS1:INT of the preceding conversion has not been cleared. To resume conversion, the ADCn_CS1:INT flag must be cleared by writing '1' to ADCn_CSC1:INTC.
	X	X	'1'	ADCn_CS1:PAUS is set after each channel conversion when the interrupt request ADCn_CS1:INT of the preceding conversion has not been cleared. To resume conversion, the ADCn_CS1:INT flag must be cleared by writing '1' to ADCn_CSC1:INTC or by reading the ADCn_CR register (except by the debug master, DAP).

Protection of ADCn_CD31~0

In continuous mode with ADCn_CS3:INTE2 = '1', ADCn_CS1:PAUS is set when data of the start channel (set by ADCn_SCH) is ready for writing to the dedicated data register (and the common data register), but ADCn_CS3:INT2 (end of scan interrupt) is already active.

Example: Start channel = 4, end channel = 7, ADCn_ER32~10:ADE[4:7] = '1111' continuous mode, ADCn_CS1:INTE = '0', and ADCn_CS3:INTE2 = '1'

Start by CPU --> convert channel 4 + save data to ADCn_CD4,

convert channel 5 + save data to ADCn_CD5,

convert channel 6 + save data to ADCn_CD6,

convert channel 7 + save data to ADCn_CD7 ---> end of scan interrupt (ADCn_CS3:INT2),

convert channel 4 + set ADCn_CS1:PAUS (protect ADCn_CD4...7).

After the CPU has read the data registers and cleared ADCn_CS3:INT2, the scan conversion continues.

This function can be disabled by setting ADCn_CS2:DPDIS = '1'. Then conversion is continued and former conversion data may be overwritten.

Protection of ADCn_CR

If ADCn_CS1:INTE = '1' or ADCn_MAR:DRQEN = '1', ADCn_CS1:PAUS is set when data of any channel is ready for writing to the common data register (and the dedicated data register) but ADCn_CS1:INT (end of conversion) is active. In this mode the protection function is active after each single conversion, the ADCn_CR register is protected.

DMA transfer

DMA transfer can be triggered by end of conversion interrupt or by end of scan interrupt provided the DMA transfer is enabled by ADCn_MAR:DRQEN and ADCn_MAR:DRQEN2 respectively.

Data protection during DMA transfer

During DMA mode of data transfer (ADCn_MAR:DRQEN = '1') the data of the ADCn_CR register is protected against overwriting if ADCn_CS2:DPDIS = '0' (data protection enabled) until the ADCn_CR register value is read (except during debug master (DAP) access). After the DMA read is over the ADCn_CS1:INT bit is internally cleared when any master except the debug master reads the register.

There is no data protection function available for the end of scan DMA request (ADCn_MAR:DRQEN2 = '1').

ADC pulse detection function

Positive events/negative events

The output of the range comparator signifies either a positive event or a negative event depending on the configuration of ADCn_RCOIRS32, ADCn_RCOIRS10 registers and the converted digital value of the ADC.

In case of range comparator setting for 'inside range', any ADC value within the range will be treated as positive event while ADC values above the upper or below the lower thresholds are treated as negative events. If the range comparator is configured for 'outside range' comparison, any ADC value within the range will be treated as negative event while ADC values above the upper or below the lower thresholds are treated as positive events. Table 36-48 shows the range comparator settings and appropriate event definition.

Table 36-48. Generation of positive/negative events

ADCn_RCOIRS32:RCOIRS, ADCn_RCOIRS10:RCOIRS value	Range comparator output	Events
'0' (configured for 'outside range')	Inside range	Negative event
	Outside range	Positive event
'1' (configured for 'inside range')	Outside range	Negative event
	Inside range	Positive event

Whenever a positive event occurs, the positive counter ADCn_PCTPCT31~0 is decremented. Similarly, whenever a corresponding negative event occurs, the corresponding negative counter ADCn_PCTNCT31~0 is decremented.

Working principle of ADC pulse detection function

Each ADC channel has its own filter with positive counter (ADCn_PCTPCT31~0), negative counter (ADCn_PCTNCT31~0), along with their reload registers (ADCn_PCTPRL31~0/ADCn_PCTNRL31~0), zero flag (ADCn_PCZF32, ADCn_PCZF10:CTPZF), and zero flag clear bit (ADCn_PCZFC32, ADCn_PCZFC10:CTPZFC). It also has the Interrupt Enable Register (ADCn_PCIE32, ADCn_PCIE10:CTPIE), the Interrupt Enable Set Register (ADCn_PCIES32, ADCn_PCIES10:CTPIES), and the Interrupt Enable Clear Register (ADCn_PCIEC32, ADCn_PCIEC10:CTPIEC).

The purpose of the positive counter is to detect consecutive ADC range comparator events of desired length. The negative counter can be used to force a restart of the positive counter if a negative signal of a certain length is detected due to spikes, noise etc.

The following steps describe the working principle.

- The positive counter ADCn_PCTPCT31~0 decrements with each positive event of the corresponding ADC channel
- The zero flag ADCn_PCZF32, ADCn_PCZF10:CTPZF is set as the positive counter reaches zero. This flag remains set until it is cleared through ADCn_PCZFC32, ADCn_PCZFC10:CTPZFC. The positive counter and the negative counter are stopped as long as zero flag of the corresponding channel is '1'
- If the zero flag (ADCn_PCZFC32, ADCn_PCZF10:CTPZF) is set and the corresponding pulse detection interrupt is enabled (ADCn_PCIE32, ADCn_PCIE10:CTPIE = '1') then an interrupt is generated
- The negative counter ADCn_PCTNCT31~0 decrements with each negative event of the corresponding ADC channel except while the corresponding zero flag is set
- The following conditions will cause a reload of the positive and/or negative counter with the values set in their reload registers ADCn_PCTPRL31~0 and ADCn_PCTNRL31~0:

  Positive counter is reloaded when

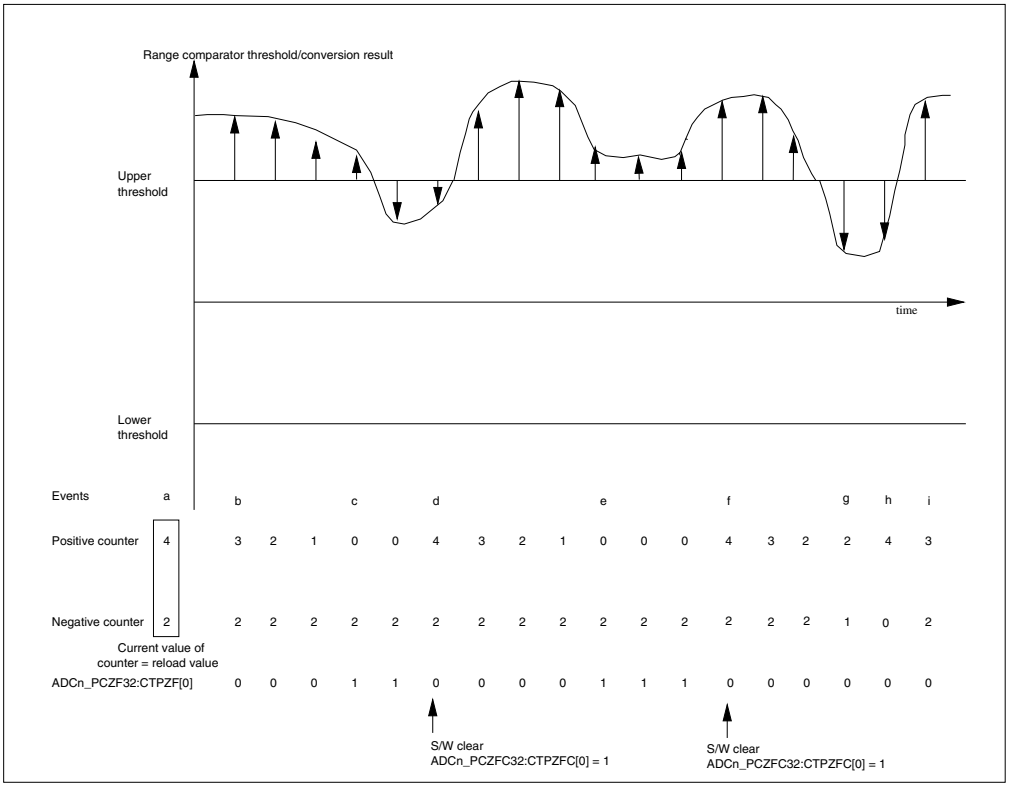
  a. Negative counter reaches zero.
  b. '1' is written to ADCn_PCZFC32, ADCn_PCZFC10:CTPZFC (independent of actual value of the corresponding ADCn_PCZF32, ADCn_PCZF10:CTPZF).

  Negative counter is reloaded when

a. Any positive event occurs.
b. '1' is written to ADCn_PCZFC32, ADCn_PCZFC10:CTPZFC (independent of actual value of the corresponding ADCn_PCZF32, ADCn_PCZF10:CTPZF).
c. The positive counter reaches zero and the zero flag is set. The negative counter will hold the reload value as long as the zero flag is not cleared.

The Figure 36-49 shows the working of the ADC pulse detection function for channel '0' with ADCn_RCOIRS32:RCOIRS[0] = '0' configured for outside range, positive reload register ADCn_PCTPRL0 = '100' and negative reload register ADCn_PCTNRL0 = '010'.

Figure 36-49. Example of ADC pulse detection function



a. Reload counters with appropriate reload value by writing ‘1’ to ADCn_PCZFC32,ADCn_PCZFC10:CTPZFC[0].
b. Positive counter decrements with positive events.
c. Positive counter expires, zero flag (ADCn_PCZF32, ADCn_PCZF10:CTPZF[0]) is set.
d. A series of negative events does not decrement the negative counter as the ADCn_PCZF32, ADCn_PCZF10:CTPZF[0] (zero flag) is set. The zero flag ADCn_PCZF32, ADCn_PCZF10:CTPZF[0] is cleared and the positive as well as the negative counter are reloaded.
e. Positive counter expires, zero flag ADCn_PCZF32, ADCn_PCZF10:CTPZF[0] = '1'.
f. S/W clear of ADCn_PCZF32, ADCn_PCZF10:CTPZF[0] reloads positive and negative counter.
g. Negative event decrements negative counter.
h. Negative counter expires and reloads positive counter.

a. Positive counter decrements with positive event.

37. Bus Support Unit

This chapter explains the functions and operations of the Bus Support Unit.

37.1 Outline of Bus Support Unit

The Bus Support Unit for a particular bus consists of a set of registers to enable/disable certain peripherals on the bus, and and to provide support for testing the bus.

Features of Bus Support Unit

The Bus Support Unit for a particular bus has the following features:

- Programmable register set to enable/disable the use of peripherals connected to the bus
- Support for testing the bus/data lines

37.2 Bus Support Unit registers

All registers of Bus Support Unit are explained in this section. The register sets are shown for Bus Support Units of all kinds of buses/peripheral groups.

37.2.1 Registers of Bus Support Unit of peripheral 0 group

The following registers are available for Bus Support Unit of peripheral 0 group

- Bus Test Low Register (BSU0_BTSTL)
- Bus Test High Register (BSU0_BTSTH)
- Peripheral Enable 0 Low Register (BSU0_PEN0L)
- Peripheral Enable 1 Low Register (BSU0_PEN1L)
- Peripheral Enable 2 Low Register (BSU0_PEN2L)
- Peripheral Enable 3 Low Register (BSU0_PEN3L)
- Peripheral Enable 4 Low Register (BSU0_PEN4L)
- Peripheral Enable 5 Low Register (BSU0_PEN5L)
- Peripheral Enable 6 Low Register (BSU0_PEN6L)
- Peripheral Enable 7 Low Register (BSU0_PEN7L)
- Peripheral Enable 7 High Register (BSU0_PEN7H)
- Peripheral Enable 8 Low Register (BSU0_PEN8L)
- Peripheral Enable 8 High Register (BSU0_PEN8H)
- Peripheral Enable 9 Low Register (BSU0_PEN9L)
- Peripheral Enable 9 High Register (BSU0_PEN9H)
- Peripheral Enable 11 Low Register (BSU0_PEN11L)

Memory layout of Bus Support Unit registers of peripheral 0 group

Table 37-1. Memory layout of Bus Support Unit registers of peripheral 0 group

Offset	+1	+0
0x00000000 - 0x00000002	reserved XXXXXXXX XXXXXXXX	
0x00000004	BSU0_BTSTL 00000000 00000000	
0x00000006	BSU0_BTSTH 00000000 00000000	
0x00000008 - 0x0000000E	reserved XXXXXXXX XXXXXXXX	
0x00000010	BSU0_PEN0L XXXXXXXX 00000000	
0x00000012	reserved XXXXXXXX XXXXXXXX	
0x00000014	BSU0_PEN1L 00000000 00000000	
0x00000016	reserved XXXXXXXX XXXXXXXX	
0x00000018	BSU0_PEN2L 00000000 00000000	
0x0000001A	reserved XXXXXXXX XXXXXXXX	
0x0000001C	BSU0_PEN3L 00000000 00000000	
0x0000001E	reserved XXXXXXXX XXXXXXXX	
0x00000020	BSU0_PEN4L XXXXXXXX 00000000	
0x00000022	reserved XXXXXXXX XXXXXXXX	
0x00000024	BSU0_PEN5L XXXXXXXX 00000000	
0x00000026	reserved XXXXXXXX XXXXXXXX	
0x00000028	BSU0_PEN6L XXXXXXXX 00000000	
0x0000002A	reserved XXXXXXXX XXXXXXXX	

Table 37-1. Memory layout of Bus Support Unit registers of peripheral 0 group

Offset	+1	+0
0x0000002C	BSU0_PEN7L 00000000 00000000	
0x0000002E	BSU0_PEN7H 00000000 00000000	
0x00000030	BSU0_PEN8L 00000000 00000000	
0x00000032	BSU0_PEN8H 00000000 00000000	
0x00000034	BSU0_PEN9L 00000000 00000000	
0x00000036	BSU0_PEN9H XXXXXXXX 00000000	
0x00000038 - 0x0000003A	reserved XXXXXXXX XXXXXXXX	
0x0000003C	BSU0_PEN11L XXXXXXXX 00000000	

37.2.1.1 Bus Test Low Register (BSU0_BTSTL)

The Bus Test Low Register contains the lower 16 bits of the Bus Test Register. The entire 32-bit BUS Test Register is rotated left by 4 bits when BSU0_BTSTH is written. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Low Register (BSU0_BTSTL)

Figure 37-1. Bus Test Low Register (BSU0_BTSTL)

BSU0_BTSTL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-2. Bus Test Low Register (BSU0_BTSTL) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BTST	Bus Test Register for Peripheral 0 Group This register holds the lower 16 bits of test vector. The content of this register is used for testing the data lines. The entire 32-bit Bus Test Register is rotated left by 4 bits when BSU0_BTSTH is written. A value written to BSU0_BTSTL and BSU0_BTSTH is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.1.2 Bus Test High Register (BSU0_BTSTH)

The Bus Test High Register contains the upper 16 bits of the Bus Test Register. The entire 32-bit BUS Test Register is rotated left by 4 bits when this register is written The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test High Register (BSU0_BTSTH)

Figure 37-2. Bus Test Register (BSU0_BTSTH)

BSU0_BTSTH															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-3. Bus Test Register (BSU0_BTSTH) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BTST	Bus Test Register for Peripheral 0 Group This register holds the higher 16 bits of test vector. The content of this register is used for testing the data lines. The entire 32-bit Bus Test Register is rotated left by 4 bits when BSU0_BTSTH is written. A value written to BSU0_BTSTL and BSU0_BTSTH is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.1.3 Peripheral Enable 0 Low Register (BSU0_PEN0L)

The Peripheral Enable 0 Low Register holds the enable bits for the peripheral 0 (ADC) of peripheral 0 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 0 Low Register (BSU0_PEN0L)

Figure 37-3. Peripheral Enable 0 Low Register (BSU0_PEN0L)

BSU0_PEN0L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENADC0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-4. Peripheral Enable 0 Low Register (BSU0_PEN0L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENADC0	Peripheral Enable bit for ADC0 '0': ADC0 is not available, '1': ADC0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.4 Peripheral Enable 1 Low Register (BSU0_PEN1L)

The Peripheral Enable 1 Low Register holds the enable bits for the peripheral 1 (FRTs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 1 Low Register (BSU0_PEN1L)

Figure 37-4. Peripheral Enable 1 Low Register (BSU0_PEN1L)

BSU0_PEN1L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENFRT3	ENFRT2	ENFRT1	ENFRT0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-5. Peripheral Enable 1 Low Register (BSU0_PEN1L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	ENFRT3	Peripheral Enable bit for FRT3 '0': FRT3 is not available '1': FRT3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENFRT2	Peripheral Enable bit for FRT2 '0': FRT2 is not available '1': FRT2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENFRT1	Peripheral Enable bit for FRT1 '0': FRT1 is not available '1': FRT1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENFRT0	Peripheral Enable bit for FRT0 '0': FRT0 is not available '1': FRT0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.5 Peripheral Enable 2 Low Register (BSU0_PEN2L)

The Peripheral Enable 2 Low Register holds the enable bits for the peripheral 2 (ICUs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 2 Low Register (BSU0_PEN2L)

Figure 37-5. Peripheral Enable 2 Low Register (BSU0_PEN2L)

BSU0_PEN2L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENICU3	ENICU2	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-6. Peripheral Enable 2 Low Register (BSU0_PEN2L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	ENICU3	Peripheral Enable bit for ICU3 '0': ICU3 is not available '1': ICU3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENICU2	Peripheral Enable bit for ICU2 '0': ICU2 is not available '1': ICU2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1:0]	read0	-

37.2.1.6 Peripheral Enable 3 Low Register (BSU0_PEN3L)

The Peripheral Enable 3 Low Register holds the enable bits for the peripheral 3 (OCUs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 3 Low Register (BSU0_PEN3L)

Figure 37-6. Peripheral Enable 3 Low Register (BSU0_PEN3L)

BSU0_PEN3L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENOCU1	ENOCU0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-7. Peripheral Enable 3 Low Register (BSU0_PEN3L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:2]	read0	-
[1]	ENOCU1	Peripheral Enable bit for OCU1 '0': OCU1 is not available '1': OCU1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENOCU0	Peripheral Enable bit for OCU0 '0': OCU0 is not available '1': OCU0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.7 Peripheral Enable 4 Low Register (BSU0_PEN4L)

The Peripheral Enable 4 Low Register holds the enable bits for the peripheral 4 (I2C) of peripheral 0 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to the lower byte even if it is an 8-bit access.

Peripheral Enable 4 Low Register (BSU0_PEN4L)

Figure 37-7. Peripheral Enable 4 Low Register (BSU0_PEN4L)

BSU0_PEN4L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENI2C0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-8. Peripheral Enable 4 Low Register (BSU0_PEN4L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENI2C0	Peripheral Enable bit for I2C0 '0': I2C0 is not available '1': I2C0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.8 Peripheral Enable 5 Low Register (BSU0_PEN5L)

The Peripheral Enable 5 Low Register holds the enable bits for the peripheral 5 (USART) of peripheral 0 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to the lower byte even if it is an 8-bit access.

Peripheral Enable 5 Low Register (BSU0_PEN5L)

Figure 37-8. Peripheral Enable 5 Low Register (BSU0_PEN5L)

BSU0_PEN5L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENUSART0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-9. Peripheral Enable 5 Low Register (BSU0_PEN5L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENUSART0	Peripheral Enable bit for USART0 '0': USART0 is not available '1': USART0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.9 Peripheral Enable 6 Low Register (BSU0_PEN6L)

The Peripheral Enable 6 Low Register holds the enable bits for the peripheral 6 (SMCs) of peripheral 0 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 6 Low Register (BSU0_PEN6L)

Figure 37-9. Peripheral Enable 6 Low Register (BSU0_PEN6L)

BSU0_PEN6L															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	ENSMCTG0	ENSMC5	ENSMC4	ENSMC3	ENSMC2	ENSMC1	ENSMC0
-	-	-	-	-	-	-	-	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-10. Peripheral Enable 6 Low Register (BSU0_PEN6L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7]	read0	-
[6]	ENSMCTG0	Peripheral Enable bit for SMC Trigger0 '0': SMC Trigger0 is not available '1': SMC Trigger0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[5]	ENSMC5	Peripheral Enable bit for SMC5 '0': SMC5 is not available '1': SMC5 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[4]	ENSMC4	Peripheral Enable bit for SMC4 '0': SMC4 is not available '1': SMC4 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

Table 37-10. Peripheral Enable 6 Low Register (BSU0_PEN6L) bits

Bit Position	Bit Field Name	Bit Description
[3]	ENSMC3	Peripheral Enable bit for SMC3 '0': SMC3 is not available '1': SMC3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENSMC2	Peripheral Enable bit for SMC2 '0': SMC2 is not available '1': SMC2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENSMC1	Peripheral Enable bit for SMC1 '0': SMC1 is not available '1': SMC1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENSMC0	Peripheral Enable bit for SMC0 '0': SMC0 is not available '1': SMC0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.10 Peripheral Enable 7 Low Register (BSU0_PEN7L)

The Peripheral Enable 7 Low Register holds the enable bits for the peripheral 7 (PPGs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 7 Low Register (BSU0_PEN7L)

Figure 37-10. Peripheral Enable 7 Low Register (BSU0_PEN7L)

BSU0_PEN7L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
ENPPG15	ENPPG14	ENPPG13	ENPPG12	ENPPG11	ENPPG10	ENPPG9	ENPPG8	ENPPG7	ENPPG6	ENPPG5	ENPPG4	ENPPG3	ENPPG2	ENPPG1	ENPPG0
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-11. Peripheral Enable 7 Low Register (BSU0_PEN7L) bits

Bit Position	Bit Field Name	Bit Description
[15]	ENPPG15	Peripheral Enable bit for PPG15 '0': PPG15 is not available '1': PPG15 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[14]	ENPPG14	Peripheral Enable bit for PPG14 '0': PPG14 is not available '1': PPG14 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[13]	ENPPG13	Peripheral Enable bit for PPG13 '0': PPG13 is not available '1': PPG13 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[12]	ENPPG12	Peripheral Enable bit for PPG12 '0': PPG12 is not available '1': PPG12 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

Table 37-11. Peripheral Enable 7 Low Register (BSU0_PEN7L) bits

Bit Position	Bit Field Name	Bit Description
[11]	ENPPG11	Peripheral Enable bit for PPG11 '0': PPG11 is not available '1': PPG11 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[10]	ENPPG10	Peripheral Enable bit for PPG10 '0': PPG10 is not available '1': PPG10 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[9]	ENPPG9	Peripheral Enable bit for PPG9 '0': PPG9 is not available '1': PPG9 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[8]	ENPPG8	Peripheral Enable bit for PPG8 '0': PPG8 is not available '1': PPG8 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[7]	ENPPG7	Peripheral Enable bit for PPG7 '0': PPG7 is not available '1': PPG7 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[6]	ENPPG6	Peripheral Enable bit for PPG6 '0': PPG6 is not available '1': PPG6 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[5]	ENPPG5	Peripheral Enable bit for PPG5 '0': PPG5 is not available '1': PPG5 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[4]	ENPPG4	Peripheral Enable bit for PPG4 '0': PPG4 is not available '1': PPG4 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

Table 37-11. Peripheral Enable 7 Low Register (BSU0_PEN7L) bits

Bit Position	Bit Field Name	Bit Description
[3]	ENPPG3	Peripheral Enable bit for PPG3 '0': PPG3 is not available '1': PPG3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENPPG2	Peripheral Enable bit for PPG2 '0': PPG2 is not available '1': PPG2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENPPG1	Peripheral Enable bit for PPG1 '0': PPG1 is not available '1': PPG1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENPPG0	Peripheral Enable bit for PPG0 '0': PPG0 is not available '1': PPG0 is available This bit is writable only once. Any additional access to this bit is ignored and results in an error condition.

37.2.1.11 Peripheral Enable 7 High Register (BSU0_PEN7H)

The Peripheral Enable 7 High Register holds the enable bits for the peripheral 7 (PPGs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 7 High Register (BSU0_PEN7H)

Figure 37-11. Peripheral Enable 7 High Register (BSU0_PEN7H)

BSU0_PEN7H															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-12. Peripheral Enable 7 High Register (BSU0_PEN7H) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	read0	-

37.2.1.12 Peripheral Enable 8 Low Register (BSU0_PEN8L)

The Peripheral Enable 8 Low Register holds the enable bits for the peripheral 8 (PPGs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 8 Low Register (BSU0_PEN8L)

Figure 37-12. Peripheral Enable 8 Low Register (BSU0_PEN8L)

BSU0_PEN8L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-13. Peripheral Enable 8 Low Register (BSU0_PEN8L) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	read0	-

37.2.1.13 Peripheral Enable 8 High Register (BSU0_PEN8H)

The Peripheral Enable 8 High Register holds the enable bits for the peripheral 8 (PPGs) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 8 High Register (BSU0_PEN8H)

Figure 37-13. Peripheral Enable 8 High Register (BSU0_PEN8H)

BSU0_PEN8H															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-14. Peripheral Enable 8 High Register (BSU0_PEN8H) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	read0	-

37.2.1.14 Peripheral Enable 9 Low Register (BSU0_PEN9L)

The Peripheral Enable 9 Low Register holds the enable bits for the peripheral 9 (PPGGC0) of peripheral 0 group. This register should be written only with 16-bit accesses.

Peripheral Enable 9 Low Register (BSU0_PEN9L)

Figure 37-14. Peripheral Enable 9 Low Register (BSU0_PEN9L)

BSU0_PEN9L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENPPGGRP3	ENPPGGRP2	ENPPGGRP1	ENPPGGRP0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	R0	R0	R0	R0	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-15. Peripheral Enable 9 Low Register (BSU0_PEN9L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-

Table 37-15. Peripheral Enable 9 Low Register (BSU0_PEN9L) bits

Bit Position	Bit Field Name	Bit Description
[3]	ENPPGGRP3	Peripheral Enable bit for PPG Group Control 3 '0': PPG group control 3 is not available '1': PPG group control 3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENPPGGRP2	Peripheral Enable bit for PPG Group Control 2 '0': PPG group control 2 is not available '1': PPG group control 2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENPPGGRP1	Peripheral Enable bit for PPG Group Control 1 '0': PPG group control 1 is not available '1': PPG group control 1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENPPGGRP0	Peripheral Enable bit for PPG Group Control 0 '0': PPG group control 0 is not available '1': PPG group control 0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.15 Peripheral Enable 9 High Register (BSU0_PEN9H)

The Peripheral Enable 9 High Register holds the enable bits for the peripheral 9 of peripheral 0 (PPGGC0) group. It is strongly recommended that this register should be written only with 16-bit access. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 9 High Register (BSU0_PEN9H)

Figure 37-15. Peripheral Enable 9 High Register (BSU0_PEN9H) bits

BSU0_PEN9H															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENPPGGLC0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-16. Peripheral Enable 9 High Register (BSU0_PEN9H) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENPPGGLC0	Peripheral Enable bit for PPG Global Control '0': PPG global control is not available '1': PPG global control is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.1.16 Peripheral Enable 11 Low Register (BSU0_PEN11L)

The Peripheral Enable 11 Low Register holds the enable bits for the peripheral 11 (BMCs) of peripheral 0 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte, even if it an 8-bit access.

Peripheral Enable 11 Low Register (BSU0_PEN11L)

Figure 37-16. Peripheral Enable 11 Low Register (BSU0_PEN11L)

BSU0_PEN11L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	read0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R0
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-17. Peripheral Enable 11 Low Register (BSU0_PEN11L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:0]	read0	-

37.2.2 Registers of Bus Support Unit of peripheral 1 group

The following registers are available for Bus Support Unit of peripheral 1 group

- Bus Test Low Register (BSU1_BTSTL)
- Bus Test High Register (BSU1_BTSTH)
- Peripheral Enable 0 Low Register (BSU1_PEN0L)
- Peripheral Enable 1 Low Register (BSU1_PEN1L)
- Peripheral Enable 2 Low Register (BSU1_PEN2L)
- Peripheral Enable 3 Low Register (BSU1_PEN3L)
- Peripheral Enable 4 Low Register (BSU1_PEN4L)
- Peripheral Enable 5 Low Register (BSU1_PEN5L)
- Peripheral Enable 6 Low Register (BSU1_PEN6L)
- Peripheral Enable 7 Low Register (BSU1_PEN7L)
- Peripheral Enable 8 Low Register (BSU1_PEN8L)
- Peripheral Enable 9 Low Register (BSU1_PEN9L)
- Peripheral Enable 9 High Register (BSU1_PEN9H)
- Peripheral Enable 10 Low Register (BSU1_PEN10L)
- Peripheral Enable 10 High Register (BSU1_PEN10H)
- Peripheral Enable 11 Low Register (BSU1_PEN11L)
- Peripheral Enable 11 High Register (BSU1_PEN11H)

Memory layout of Bus Support Unit registers of peripheral 1 group

Table 37-18. Memory layout of Bus Support Unit registers of peripheral 1 group

Offset	+1	+0
0x00000000 - 0x00000002	reserved XXXXXXXX XXXXXXXX	
0x00000004	BSU1_BTSTL 00000000 00000000	
0x00000006	BSU1_BTSTH 00000000 00000000	
0x00000008 - 0x0000000E	reserved XXXXXXXX XXXXXXXX	
0x00000010	BSU1_PEN0L XXXXXXXX 00000000	
0x00000012	reserved XXXXXXXX XXXXXXXX	
0x00000014	BSU1_PEN1L XXXXXXXX 00000000	
0x00000016	reserved XXXXXXXX XXXXXXXX	
0x00000018	BSU1_PEN2L XXXXXXXX 00000000	

Table 37-18. Memory layout of Bus Support Unit registers of peripheral 1 group

Offset	+1	+0
0x0000001A	reserved XXXXXXXX XXXXXXXX	
0x0000001C	BSU1_PEN3L 00000000 00000000	
0x0000001E	reserved XXXXXXXX XXXXXXXX	
0x00000020	BSU1_PEN4L 00000000 00000000	
0x00000022	reserved XXXXXXXX XXXXXXXX	
0x00000024	BSU1_PEN5L 00000000 00000000	
0x00000026	reserved XXXXXXXX XXXXXXXX	
0x00000028	BSU1_PEN6L XXXXXXXX 00000000	
0x0000002A	reserved XXXXXXXX XXXXXXXX	
0x0000002C	BSU1_PEN7L XXXXXXXX 00000000	
0x0000002E	reserved XXXXXXXX XXXXXXXX	
0x00000030	BSU1_PEN8L XXXXXXXX 00000000	
0x00000032	reserved XXXXXXXX XXXXXXXX	
0x00000034	BSU1_PEN9L 00000000 00000000	
0x00000036	BSU1_PEN9H 00000000 00000000	
0x00000038	BSU1_PEN10L 00000000 00000000	
0x0000003A	BSU1_PEN10H 00000000 00000000	
0x0000003C	BSU1_PEN11L 00000000 00000000	
0x0000003E	BSU1_PEN11H XXXXXXXX 00000000	

37.2.2.1 Bus Test Low Register (BSU1_BTSTL)

The Bus Test Low Register contains the lower 16 bits of the Bus Test Register. The entire 32-bit BUS Test Register is rotated left by 4 bits when BSU1_BTSTH is written The value from this register can be read back for test the data lines of the bus and to check whether it matches with the expected data which has been rorated and written.

Bus Test Low Register (BSU1_BTSTL)

Figure 37-17. Bus Test Low Register (BSU1_BTSTL)

BSU1_BTSTL															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-19. Bus Test Low Register (BSU1_BTSTL) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BTST	Bus Test Register for Peripheral 1 Group This register holds the lower 16 bits of test vector. The contents of this register are used for testing the data line. The entire 32-bit Bus Test Register is rotated left by 5 bits when BSU1_BTSTH is written. A value written to BSU1_BTSTL and BSU1_BTSTH is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.2.2 Bus Test High Register (BSU1_BTSTH)

The Bus Test High Register contains the upper 16 bits of the Bus Test Register. The entire 32-bit BUS Test Register is rotated left by 5 bits when this register is written. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test High Register (BSU1_BTSTH)

Figure 37-18. Bus Test High Register (BSU1_BTSTH)

BSU1_BTSTH															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-20. Bus Test High Register (BSU1_BTSTH) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BTST	Bus Test Register for Peripheral 1 Group This register holds the higher 16 bits of test vector. The contents of this register are used for testing the data line. The entire 32-bit Bus Test Register is rotated left by 5 bits when this register is written. A value written to BSU1_BTSTL and BSU1_BTSTH is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.2.3 Peripheral Enable 0 Low Register (BSU1_PEN0L)

The Peripheral Enable 0 Low Register holds the enable bits for the peripheral 0 (SGs) of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte, even if it an 8-bit access.

Peripheral Enable 0 Low Register (BSU1_PEN0L)

Figure 37-19. Peripheral Enable 0 Low Register (BSU1_PEN0L)

BSU1_PEN0L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENSG0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-21. Peripheral Enable 0 Low Register (BSU1_PEN0L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENSG0	Peripheral Enable bit for Sound Generator 0 '0': Sound Generator 0 is not available '1': Sound Generator 0 is available This bit is writable only once. Any additional access to this bit is ignored and results in an error condition.

37.2.2.4 Peripheral Enable 1 Low Register (BSU1_PEN1L)

The Peripheral Enable 1 Low Register holds the enable bits for the peripheral 1 (CANs) of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 1 Low Register (BSU1_PEN1L)

Figure 37-20. Peripheral Enable 1 Low Register (BSU1_PEN1L)

BSU1_PEN1L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	ENCAN2	ENCAN1	ENCAN0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	RWP	RWP	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-22. Peripheral Enable 1 Low Register (BSU1_PEN1L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:3]	read0	-
[2]	ENCAN2	Peripheral Enable bit for CAN2 '0': CAN2 is not available '1': CAN2 is available This bit is writable only once. Any additional access to this bit is ignored and results in an error condition.
[1]	ENCAN1	Peripheral Enable bit for CAN1 '0': CAN1 is not available '1': CAN1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENCAN0	Peripheral Enable bit for CAN0 '0': CAN0 is not available '1': CAN0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.2.5 Peripheral Enable 2 Low Register (BSU1_PEN2L)

The Peripheral Enable 2 Low Register holds the enable bits for the peripheral 2 (LCD) of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 2 Low Register (BSU1_PEN2L)

Figure 37-21. Peripheral Enable 2 Low Register (BSU1_PEN2L)

BSU1_PEN2L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	read0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R0
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-23. Peripheral Enable 2 Low Register (BSU1_PEN2L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:0]	read0	-

37.2.2.6 Peripheral Enable 3 Low Register (BSU1_PEN3L)

The Peripheral Enable 3 Low Register holds the enable bits for the peripheral 3 (FRTs) of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 3 Low Register (BSU1_PEN3L)

Figure 37-22. Peripheral Enable 3 Low Register (BSU1_PEN3L)

BSU1_PEN3L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENFRT19	ENFRT18	ENFRT17	ENFRT16
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-24. Peripheral Enable 3 Low Register (BSU1_PEN3L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	ENFRT19	Peripheral Enable bit for FRT19 '0': FRT19 is not available '1': FRT19 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENFRT18	Peripheral Enable bit for FRT18 '0': FRT18 is not available '1': FRT18 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENFRT17	Peripheral Enable bit for FRT17 '0': FRT17 is not available '1': FRT17 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENFRT16	Peripheral Enable bit for FRT16 '0': FRT16 is available '1': FRT16 is not available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.2.7 Peripheral Enable 4 Low Register (BSU1_PEN4L)

The Peripheral Enable 4 Low Register holds the enable bits for the peripheral 4 (ICUs) of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 4 Low Register (BSU1_PEN4L)

Figure 37-23. Peripheral Enable 4 Low Register (BSU1_PEN4L)

BSU1_PEN4L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENICU19	ENICU18	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-25. Peripheral Enable 4 Low Register (BSU1_PEN4L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:4]	read0	-
[3]	ENICU19	Peripheral Enable bit for ICU19 '0': ICU19 is not available '1': ICU19 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENICU18	Peripheral Enable bit for ICU18 '0': ICU18 is not available '1': ICU18 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1:0]	read0	-

37.2.2.8 Peripheral Enable 5 Low Register (BSU1_PEN5L)

The Peripheral Enable 5 Low Register holds the enable bits for the peripheral 5 (OCUs) of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 5 Low Register (BSU1_PEN5L)

Figure 37-24. Peripheral Enable 5 Low Register (BSU1_PEN5L)

BSU1_PEN5L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENOCU17	ENOCU16
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-26. Peripheral Enable 5 Low Register (BSU1_PEN5L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:2]	read0	-
[1]	ENOCU17	Peripheral Enable bit for OCU17 '0': OCU17 is not available '1': OCU17 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENOCU16	Peripheral Enable bit for OCU16 '0': OCU16 is not available '1': OCU16 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.2.9 Peripheral Enable 6 Low Register (BSU1_PEN6L)

The Peripheral Enable 6 Low Register holds the enable bits for the peripheral 6 of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 6 Low Register (BSU1_PEN6L)

Figure 37-25. Peripheral Enable 6 Low Register (BSU1_PEN6L)

BSU1_PEN6L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	read0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R0
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-27. Peripheral Enable 6 Low Register (BSU1_PEN6L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:0]	read0	-

37.2.2.10 Peripheral Enable 7 Low Register (BSU1_PEN7L)

The Peripheral Enable 7 Low Register holds the enable bits for the peripheral 7 (USART) of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 7 Low Register (BSU1_PEN7L)

Figure 37-26. Peripheral Enable 7 Low Register (BSU1_PEN7L)

BSU1_PEN7L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENUSART6
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-28. Peripheral Enable 7 Low Register (BSU1_PEN7L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENUSART6	Peripheral Enable bit for USART6 '0': USART6 is not available '1': USART6 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.2.11 Peripheral Enable 8 Low Register (BSU1_PEN8L)

The Peripheral Enable 8 Low Register holds the enable bits for the peripheral 8 (SMCs) of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 8 Low Register (BSU1_PEN8L)

Figure 37-27. Peripheral Enable 8 Low Register (BSU1_PEN8L)

BSU1_PEN8L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	read0
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R0
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-29. Peripheral Enable 8 Low Register (BSU1_PEN8L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:0]	read0	-

37.2.2.12 Peripheral Enable 9 Low Register (BSU1_PEN9L)

The Peripheral Enable 9 Low Register holds the enable bits for the peripheral 9 (PPGs) of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 9 Low Register (BSU1_PEN9L)

Figure 37-28. Peripheral Enable 9 Low Register (BSU1_PEN9L)

BSU1_PEN9L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	ENPPG71	ENPPG70	ENPPG69	ENPPG68	ENPPG67	ENPPG66	ENPPG65	ENPPG64
R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-30. Peripheral Enable 9 Low Register (BSU1_PEN9L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7]	ENPPG71	Peripheral Enable bit for PPG71 '0': PPG71 is not available '1': PPG71 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[6]	ENPPG70	Peripheral Enable bit for PPG70 '0': PPG70 is not available '1': PPG70 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[5]	ENPPG69	Peripheral Enable bit for PPG69 '0': PPG69 is not available '1': PPG69 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[4]	ENPPG68	Peripheral Enable bit for PPG68 '0': PPG68 is not available '1': PPG68 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

Table 37-30. Peripheral Enable 9 Low Register (BSU1_PEN9L) bits

Bit Position	Bit Field Name	Bit Description
[3]	ENPPG67	Peripheral Enable bit for PPG 67 '0': PPG67 is not available '1': PPG67 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENPPG66	Peripheral Enable bit for PPG66 '0': PPG66 is not available '1': PPG66 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENPPG65	Peripheral Enable bit for PPG65 '0': PPG65 is not available '1': PPG65 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENPPG64	Peripheral Enable bit for PPG64 '0': PPG64 is not available '1': PPG64 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.2.13 Peripheral Enable 9 High Register (BSU1_PEN9H)

The Peripheral Enable 9 High Register holds the enable bits for the peripheral 9 of Peripheral 1 group. This register should be written only with 16 bit accesses.

Peripheral Enable 9 High Register (BSU1_PEN9H)

Figure 37-29. Peripheral Enable 9 High Register (BSU1_PEN9H)

BSU1_PEN9H															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-31. Peripheral Enable 9 High Register (BSU1_PEN9H) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	read0	-

37.2.2.14 Peripheral Enable 10 Low Register (BSU1_PEN10L)

The Peripheral Enable 10 Low Register holds the enable bits for the peripheral 10 of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 10 Low Register (BSU1_PEN10L)

Figure 37-30. Peripheral Enable 10 Low Register (BSU1_PEN10L)

BSU1_PEN10L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-32. Peripheral Enable 10 Low Register (BSU1_PEN10L) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	read0	-

37.2.2.15 Peripheral Enable 10 High Register (BSU1_PEN10H)

The Peripheral Enable 10 High Register holds the enable bits for the peripheral 10 of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 10 High Register (BSU1_PEN10H)

Figure 37-31. Peripheral Enable 10 High Register (BSU1_PEN10H)

BSU1_PEN10H															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-33. Peripheral Enable 10 High Register (BSU1_PEN10H) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	read0	-

37.2.2.16 Peripheral Enable 11 Low Register (BSU1_PEN11L)

The Peripheral Enable 11 Low Register holds the enable bits for the peripheral 11 (PPGGC1) of peripheral 1 group. This register should be written only with 16-bit accesses.

Peripheral Enable 11 Low Register (BSU1_PEN11L)

Figure 37-32. Peripheral Enable 11 Low Register (BSU1_PEN11L)

BSU1_PEN11L															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENPPGGRP17	ENPPGGRP16
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-34. Peripheral Enable 11 Low Register (BSU1_PEN11L) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	read0	-
[7:2]	read0	-
[1]	ENPPGGRP17	Peripheral Enable bit for PPG Group Control 17 '0': PPG group control 17 is not available '1': PPG group control 17 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENPPGGRP16	Peripheral Enable bit for PPG Group Control 16 '0': PPG group control 16 is not available '1': PPG group control 16 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.2.17 Peripheral Enable 11 High Register (BSU1_PEN11H)

The Peripheral Enable 11 High Register holds the enable bits for the peripheral 11 (PPGGC1) of peripheral 1 group. It is strongly recommended that this register should be written only with 16-bit accesses. However, an error response/BECU error will not be generated for the first write access to lower byte even if it is an 8-bit access.

Peripheral Enable 11 High Register (BSU1_PEN11H)

Figure 37-33. Peripheral Enable 11 High Register (BSU1_PEN11H)

BSU1_PEN11H															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENPPGGLC1
-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-35. Peripheral Enable 11 High Register (BSU1_PEN11H) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	ENPPGGLC1	Peripheral Enable bit for PPG Global Control 1 '0': PPG Global control 1 is not available '1': PPG Global control 1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.3 Registers of Bus Support Unit of peripheral 3 group

The following registers are available for Bus Support Unit of peripheral 3 group

- Bus Test Register (BSU3_BTST)
- Peripheral Enable 2 Register (BSU3_PEN2)
- Peripheral Enable 4 Register (BSU3_PEN4)

Memory layout of Bus Support Unit registers of peripheral 3 group

Table 37-36. Memory layout of Bus Support Unit registers of peripheral 3 group

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	BSU3_BTST 00000000 00000000 00000000 00000000				reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000008 - 0x00000010	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000018	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				BSU3_PEN2 00000000 00000000 00000000 00000000			
0x00000020	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				BSU3_PEN4 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			

37.2.3.1 Bus Test Register (BSU3_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 7 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU3_BTST)

Figure 37-34. Bus Test Register (BSU3_BTST)

BSU3_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-37. Bus Test Register (BSU3_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register for Peripheral 3 Group This register holds the value of test vec- tor. The contents of this register are used for testing the data line. A value written to BSU3_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.3.2 Peripheral Enable 2 Register (BSU3_PEN2)

The Peripheral Enable 2 Register holds the enable bits for the peripheral 2 (RLTs) of peripheral 3 group. This register should be written only with 32-bit accesses.

Peripheral Enable 2 Register (BSU3_PEN2)

Figure 37-35. Peripheral Enable 2 Register (BSU3_PEN2)

BSU3_PEN2																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENRLT9	ENRLT8	ENRLT7	ENRLT6	ENRLT5	ENRLT4	ENRLT3	ENRLT2	ENRLT1	ENRLT0
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-38. Peripheral Enable 2 Register (BSU3_PEN2) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	ENRLT9	Peripheral Enable bit for RLT9 '0': RLT9 is not available '1': RLT9 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[8]	ENRLT8	Peripheral Enable bit for RLT8 '0': RLT8 is not available '1': RLT8 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[7]	ENRLT7	Peripheral Enable bit for RLT7 '0': RLT7 is not available '1': RLT7 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[6]	ENRLT6	Peripheral Enable bit for RLT6 '0': RLT6 is not available '1': RLT6 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

Table 37-38. Peripheral Enable 2 Register (BSU3_PEN2) bits

Bit Position	Bit Field Name	Bit Description
[5]	ENRLT5	Peripheral Enable bit for RLT5 '0': RLT5 is not available '1': RLT5 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[4]	ENRLT4	Peripheral Enable bit for RLT4 '0': RLT4 is not available '1': RLT4 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[3]	ENRLT3	Peripheral Enable bit for RLT3 '0': RLT3 is not available '1': RLT3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[2]	ENRLT2	Peripheral Enable bit for RLT2 '0': RLT2 is not available '1': RLT2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENRLT1	Peripheral Enable bit for RLT1 '0': RLT1 is not available '1': RLT1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENRLT0	Peripheral Enable bit for RLT0 '0': RLT0 is not available '1': RLT0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.3.3 Peripheral Enable 4 Register (BSU3_PEN4)

The Peripheral Enable 4 Register holds the enable bits for the peripheral 4 (UDC) of peripheral 3 group. This register should be written only with 32-bit accesses.

Peripheral Enable 4 Register (BSU3_PEN4)

Figure 37-36. Peripheral Enable 4 Register (BSU3_PEN4)

BSU3_PEN4																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENUDC0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-39. Peripheral Enable 4 Register (BSU3_PEN4) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENUDC0	Peripheral Enable bit for UDC0 '0': UDC0 is not available '1': UDC0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.4 Registers of Bus Support Unit of peripheral 4 slave group

The following registers are available for Bus Support Unit of peripheral 4 slave group

- Bus Test Register (BSU4_BTST)
- Peripheral Enable 1 Register (BSU4_PEN1)
- Peripheral Enable 2 Register (BSU4_PEN2)
- Peripheral Enable 3 Register (BSU4_PEN3)
- Peripheral Enable 4 Register (BSU4_PEN4)
- Peripheral Enable 5 Register (BSU4_PEN5)
- Peripheral Enable 6 Register (BSU4_PEN6)
- Peripheral Enable 7 Register (BSU4_PEN7)
- Peripheral Enable 8 Register (BSU4_PEN8)

Memory layout of Bus Support Unit registers of peripheral 4 slave group

Table 37-40. Memory layout of Bus Support Unit registers of peripheral 4 slave group

Offset	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000004	BSU4_BTST 00000000 00000000 00000000 00000000			
0x00000008 - 0x00000010	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000014	BSU4_PEN1 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000018	BSU4_PEN2 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x0000001C	BSU4_PEN3 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000020	BSU4_PEN4 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000024	BSU4_PEN5 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000028	BSU4_PEN6 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x0000002C	BSU4_PEN7 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000030	BSU4_PEN8 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			

37.2.4.1 Bus Test Register (BSU4_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 8 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU4_BTST)

Figure 37-37. Bus Test Register (BSU4_BTST)

BSU4_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-41. Bus Test Register (BSU4_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register for Peripheral 4 Slave Group This register holds the value of test vector. The contents of this register are used for testing the data line. A value written to BSU4_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.4.2 Peripheral Enable 1 Register (BSU4_PEN1)

The Peripheral Enable 1 Register holds the enable bits for the peripheral 1 (ETH) of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 1 Register (BSU4_PEN1)

Figure 37-38. Peripheral Enable 1 Register (BSU4_PEN1)

BSU4_PEN1																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENETH0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-42. Peripheral Enable 1 Register (BSU4_PEN1) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENETH0	Peripheral Enable bit for ETH0 '0': ETH0 is not available '1': ETH0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.4.3 Peripheral Enable 2 Register (BSU4_PEN2)

The Peripheral Enable 2 Register holds the enable bits for the peripheral 2 (MLB) of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 2 Register (BSU4_PEN2)

Figure 37-39. Peripheral Enable 2 Register (BSU4_PEN2)

BSU4_PEN2																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENMLB0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-43. Peripheral Enable 2 Register (BSU4_PEN2) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENMLB0	Peripheral Enable bit for MLB0 '0': MLB0 is not available '1': MLB0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.4.4 Peripheral Enable 3 Register (BSU4_PEN3)

The Peripheral Enable 3 Register holds the enable bits for the peripheral 3 of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 3 Register (BSU4_PEN3)

Figure 37-40. Peripheral Enable 3 Register (BSU4_PEN3)

BSU4_PEN3																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	read0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-44. Peripheral Enable 3 Register (BSU4_PEN3) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:0]	read0	-

37.2.4.5 Peripheral Enable 4 Register (BSU4_PEN4)

The Peripheral Enable 4 Register holds the enable bits for the peripheral 4 (I2Ss) of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 4 Register (BSU4_PEN4)

Figure 37-41. Peripheral Enable 4 Register (BSU4_PEN4)

BSU4_PEN4																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	ENI2S1	ENI2S0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	RWP	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-45. Peripheral Enable 4 Register (BSU4_PEN4) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:2]	read0	-
[1]	ENI2S1	Peripheral Enable bit for I2S1 '0': I2S1 is not available '1': I2S1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENI2S0	Peripheral Enable bit for I2S0 '0': I2S0 is not available '1': I2S0 is available This bit is writable only once. Any additional access to this bit is ignored and results in an error condition.

37.2.4.6 Peripheral Enable 5 Register (BSU4_PEN5)

The Peripheral Enable 5 Register holds the enable bits for the peripheral 5 of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 5 Register (BSU4_PEN5)

Figure 37-42. Peripheral Enable 5 Register (BSU4_PEN5)

BSU4_PEN5																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	read0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-46. Peripheral Enable 5 Register (BSU4_PEN5) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:0]	read0	-

37.2.4.7 Peripheral Enable 6 Register (BSU4_PEN6)

The Peripheral Enable 6 Register holds the enable bits for the peripheral 6 (CRC) of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 6 Register (BSU4_PEN6)

Figure 37-43. Peripheral Enable 6 Register (BSU4_PEN6)

BSU4_PEN6																													
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
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENCRC0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-47. Peripheral Enable 6 Register (BSU4_PEN6) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENCRC0	Peripheral Enable bit for CRC0 '0': CRC0 is not available '1': CRC0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.4.8 Peripheral Enable 7 Register (BSU4_PEN7)

The Peripheral Enable 7 Register holds the enable bits for the peripheral 7 (SPIs) of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 7 Register (BSU4_PEN7)

Figure 37-44. Peripheral Enable 7 Register (BSU4_PEN7)

BSU4_PEN7																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	ENSPI2	ENSPI1	ENSPI0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	RWP	RWP	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-48. Peripheral Enable 7 Register (BSU4_PEN7) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:3]	read0	-
[2]	ENSPI2	Peripheral Enable bit for SPI2 '0': SPI2 is not available '1': SPI2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[1]	ENSPI1	Peripheral Enable bit for SPI1 '0': SPI1 is not available '1': SPI1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[0]	ENSPI0	Peripheral Enable bit for SPI0 '0': SPI0 is not available '1': SPI0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.4.9 Peripheral Enable 8 Register (BSU4_PEN8)

The Peripheral Enable 8 Register holds the enable bits for the peripheral 8 (ARH) of peripheral 4 group. This register should be written only with 32-bit accesses.

Peripheral Enable 8 Register (BSU4_PEN8)

Figure 37-45. Peripheral Enable 8 Register (BSU4_PEN8)

BSU4_PEN8																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENARH0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-49. Peripheral Enable 8 Register (BSU4_PEN8) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENARH0	Peripheral Enable bit for ENARH0 '0': ARH0 is not available '1': ARH0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.5 Registers of Bus Support Unit of peripheral 5 group

The following register is available for Bus Support Unit of peripheral 5 group

- Bus Test Register (BSU5_BTST)

Memory layout of Bus Support Unit registers of Peripheral 5 group

Table 37-50. Memory layout of Bus Support Unit registers of Peripheral 5 group

Offset	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000004	BSU5_BTST 00000000 00000000 00000000 00000000			

37.2.5.1 Bus Test Register (BSU5_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 9 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU5_BTST)

Figure 37-46. Bus Test Register (BSU5_BTST)

BSU5_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-51. Bus Test Register (BSU5_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register for Peripheral 5 Group This register holds the value of test vec- tor. The contents of this register are used for testing the data line. A value written to BSU5_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.6 Registers of Bus Support Unit of MEMORY_CONFIG group

The following registers are available for Bus Support Unit of MEMORY_CONFIG group

- Bus Test Register (BSU6_BTST)
- Peripheral Enable 2 Register (BSU6_PEN2)
- Peripheral Enable 4 Register (BSU6_PEN4)
- Peripheral Enable 12 Register (BSU6_PEN12)
- Peripheral Enable 20 Register (BSU6_PEN20)

Memory layout of Bus Support Unit registers of MEMORY_CONFIG group

Table 37-52. Memory layout of Bus Support Unit registers of MEMORY_CONFIG group

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	BSU6_BTST 00000000 00000000 00000000 00000000				reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000008 - 0x00000010	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000018	reserved_5 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				BSU6_PEN2 XXXXXXXX XXXXXXXX 00000001 XXXXXXXX			
0x00000020	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				BSU6_PEN4 XXXXXXXX XXXXXXXX XXXXXXXX 00000001			
0x00000028 - 0x00000038	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000040	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				BSU6_PEN12 XXXXXXXX XXXXXXXX XXXXXXXX 00000001			
0x00000048 - 0x00000058	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX							
0x00000060	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				BSU6_PEN20 XXXXXXXX XXXXXXXX XXXXXXXX 00000001			

37.2.6.1 Bus Test Register (BSU6_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 11 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU6_BTST)

Figure 37-47. Bus Test Register (BSU6_BTST)

BSU6_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-53. Bus Test Register (BSU6_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register This register holds the value of test vector. The contents of this register are used for testing the data line. A value written to BSU6_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.6.2 Peripheral Enable 2 Register (BSU6_PEN2)

The Peripheral Enable 2 Register holds the enable bits for the peripheral 2 (EEFCFG) of the MEMORY_CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 2 Register (BSU6_PEN2)

Figure 37-48. Peripheral Enable 2 Register (BSU6_PEN2)

BSU6_PEN2																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENEEFCFG	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP	-	-	-	-	-	-	-	-
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X

Table 37-54. Peripheral Enable 2 Register (BSU6_PEN2) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	reserved	-
[15:9]	read0	-
[8]	ENEEFCFG	Peripheral Enable bit for EEFCFG '0': EEFCFG is not available '1': EEFCFG is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[7:0]	reserved	-

37.2.6.3 Peripheral Enable 4 Register (BSU6_PEN4)

The Peripheral Enable 4 Register holds the enable bits for the peripheral 4 (EEFLASH mirror area 0) of the MEMORY_CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 4 Register (BSU6_PEN4)

Figure 37-49. Peripheral Enable 4 Register (BSU6_PEN4)

BSU6_PEN4																													
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
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENEEFLASHMIR0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 37-55. Peripheral Enable 4 Register (BSU6_PEN4) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENEEFLASHMIR0	Peripheral Enable bit for EEFLASH Mirror Area 0 '0': EEFLASHMIR0 is not available '1': EEFLASHMIR0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.6.4 Peripheral Enable 12 Register (BSU6_PEN12)

The Peripheral Enable 12 Register holds the enable bits for the peripheral 12 (EEFLASH mirror area 1) of the MEMORY_-CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 12 Register (BSU6_PEN12)

Figure 37-50. Peripheral Enable 12 Register (BSU6_PEN12)

BSU6_PEN12																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENEEFLASHMIR1
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 37-56. Peripheral Enable 12 Register (BSU6_PEN12) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENEEFLASHMIR1	Peripheral Enable bit for EEFLASH Mirror Area 1 '0': EEFLASHMIR1 is not available '1': EEFLASHMIR1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.6.5 Peripheral Enable 20 Register (BSU6_PEN20)

The Peripheral Enable 20 Register holds the enable bits for the peripheral 20 (EEFLASH mirror area 2) of the MEMORY_-CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 20 Register (BSU6_PEN20)

Figure 37-51. Peripheral Enable 20 Register (BSU6_PEN20)

BSU6_PEN20																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENEEFLASHMIR2
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 37-57. Peripheral Enable 20 Register (BSU6_PEN20) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENEEFLASHMIR2	Peripheral Enable bit for EEFLASH Mirror Area 2 '0': EEFLASHMIR2 is not available '1': EEFLASHMIR2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.7 Registers of Bus Support Unit of the MCU_CONFIG group

The following registers are available for Bus Support Unit of the MCU_CONFIG group

- Bus Test Register (BSU7_BTST)
- Peripheral Enable 3 Register (BSU7_PEN3)
- Peripheral Enable 5 Register (BSU7_PEN5)
- Peripheral Enable 7 Register (BSU7_PEN7)
- Peripheral Enable 8 Register (BSU7_PEN8)

Memory layout of Bus Support Unit registers of MCU_CONFIG group

Table 37-58. Memory layout of Bus Support Unit registers of MCU_CONFIG group

Offset	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000004	BSU7_BTST 00000000 00000000 00000000 00000000			
0x00000008 - 0x00000018	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000001C	BSU7_PEN3 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000020	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000024	BSU7_PEN5 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			
0x00000028	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000002C	BSU7_PEN7 00000000 00000000 00000000 00000000			
0x00000030	BSU7_PEN8 00000000 00000000 00000000 00000000			

37.2.7.1 Bus Test Register (BSU7_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 10 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU7_BTST)

Figure 37-52. Bus Test Register (BSU7_BTST)

BSU7_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-59. Bus Test Register (BSU7_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register This register holds the value of test vector. The contents of this register are used for testing the data line. A value written to BSU7_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.7.2 Peripheral Enable 3 Register (BSU7_PEN3)

The Peripheral Enable 3 Register holds the enable bits for the peripheral 3 (RTC) of MCU_CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 3 Register (BSU7_PEN3)

Figure 37-53. Peripheral Enable 3 Register (BSU7_PEN3)

BSU7_PEN3																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENRTC
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-60. Peripheral Enable 3 Register (BSU7_PEN3) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENRTC	Peripheral Enable bit for RTC '0': RTC is not available '1': RTC is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.7.3 Peripheral Enable 5 Register (BSU7_PEN5)

The Peripheral Enable 5 Register holds the enable bits for the peripheral 5 (EICU) of the MCU_CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 5 Register (BSU7_PEN5)

Figure 37-54. Peripheral Enable 5 Register (BSU7_PEN5)

BSU7_PEN5																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENEICU0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-61. Peripheral Enable 5 Register (BSU7_PEN5) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENEICU0	Peripheral Enable bit for EICU0 '0': EICU0 is not available '1': EICU0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.7.4 Peripheral Enable 7 Register (BSU7_PEN7)

The Peripheral Enable 7 Register holds the enable bits for the peripheral 7 (RR bank 0 to 7) of the MCU_CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 7 Register (BSU7_PEN7)

Figure 37-55. Peripheral Enable 7 Register (BSU7_PEN7)

BSU7_PEN7																															
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**
read0	read0	read0	ENRETRAMBANK7	read0	read0	read0	ENRETRAMBANK6	read0	read0	read0	ENRETRAMBANK5	read0	read0	read0	ENRETRAMBANK4	read0	read0	read0	ENRETRAMBANK3	read0	read0	read0	ENRETRAMBANK2	read0	read0	read0	ENRETRAMBANK1	read0	read0	read0	ENRETRAMBANK0
R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-62. Peripheral Enable 7 Register (BSU7_PEN7) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28]	ENRETRAMBANK7	Peripheral Enable bit for RETRAMBANK7 '0': RETRAMBANK7 is not available '1': RETRAMBANK7 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[27:25]	read0	-
[24]	ENRETRAMBANK6	Peripheral Enable bit for RETRAMBANK6 '0': RETRAMBANK6 is not available '1': RETRAMBANK6 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[23:21]	read0	-
[20]	ENRETRAMBANK5	Peripheral Enable bit for RETRAMBANK5 '0': RETRAMBANK5 is not available '1': RETRAMBANK5 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[19:17]	read0	-

Table 37-62. Peripheral Enable 7 Register (BSU7_PEN7) bits

Bit Position	Bit Field Name	Bit Description
[16]	ENRETRAMBANK4	Peripheral Enable bit for RETRAMBANK4 '0': RETRAMBANK4 is not available '1': RETRAMBANK4 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[15:13]	read0	-
[12]	ENRETRAMBANK3	Peripheral Enable bit for RETRAMBANK3 '0': RETRAMBANK3 is not available '1': RETRAMBANK3 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[11:9]	read0	-
[8]	ENRETRAMBANK2	Peripheral Enable bit for RETRAMBANK2 '0': RETRAMBANK2 is not available '1': RETRAMBANK2 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[7:5]	read0	-
[4]	ENRETRAMBANK1	Peripheral Enable bit for RETRAMBANK1 '0': RETRAMBANK1 is not available '1': RETRAMBANK1 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[3:1]	read0	-
[0]	ENRETRAMBANK0	Peripheral Enable bit for RETRAMBANK0 '0': RETRAMBANK0 is not available '1': RETRAMBANK0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.7.5 Peripheral Enable 8 Register (BSU7_PEN8)

The Peripheral Enable 8 Register holds the enable bits for the peripheral 8 (RR bank 8 to 15) of the MCU_CONFIG group. This register should be written only with 32-bit accesses.

Peripheral Enable 8 Register (BSU7_PEN8)

Figure 37-56. Peripheral Enable 8 Register (BSU7_PEN8)

BSU7_PEN8																															
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**
read0	read0	read0	ENRETRAMBANK15	read0	read0	read0	ENRETRAMBANK14	read0	read0	read0	ENRETRAMBANK13	read0	read0	read0	ENRETRAMBANK12	read0	read0	read0	ENRETRAMBANK11	read0	read0	read0	ENRETRAMBANK10	read0	read0	read0	ENRETRAMBANK9	read0	read0	read0	ENRETRAMBANK8
R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP	R0	R0	R0	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-63. Peripheral Enable 8 Register (BSU7_PEN8) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28]	ENRETRAMBANK15	Peripheral Enable bit for RETRAMBANK15 '0': RETRAMBANK15 is not available '1': RETRAMBANK15 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[27:25]	read0	-
[24]	ENRETRAMBANK14	Peripheral Enable bit for RETRAMBANK14 '0': RETRAMBANK14 is not available '1': RETRAMBANK14 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[23:21]	read0	-
[20]	ENRETRAMBANK13	Peripheral Enable bit for RETRAMBANK13 '0': RETRAMBANK13 is not available '1': RETRAMBANK13 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[19:17]	read0	-

Table 37-63. Peripheral Enable 8 Register (BSU7_PEN8) bits

Bit Position	Bit Field Name	Bit Description
[16]	ENRETRAMBANK12	Peripheral Enable bit for RETRAMBANK12 '0': RETRAMBANK12 is not available '1': RETRAMBANK12 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[15:13]	read0	-
[12]	ENRETRAMBANK11	Peripheral Enable bit for RETRAMBANK11 '0': RETRAMBANK11 is not available '1': RETRAMBANK11 is available This bit is writable only once. Any additional rite access to this bit is ignored and results in an error condition.
[11:9]	read0	-
[8]	ENRETRAMBANK10	Peripheral Enable bit for RETRAMBANK10 '0': RETRAMBANK10 is not available '1': RETRAMBANK10 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[7:5]	read0	-
[4]	ENRETRAMBANK9	Peripheral Enable bit for RETRAMBANK9 '0': RETRAMBANK9 is not available '1': RETRAMBANK9 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.
[3:1]	read0	-
[0]	ENRETRAMBANK8	Peripheral Enable bit for RETRAMBANK8 '0': RETRAMBANK8 is not available '1': RETRAMBANK8 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.8 Registers of Bus Support Unit of the HSSPI0 group

Memory layout of Bus Support Unit registers of the HSSPI0 group

- Bus Test Register (BSU8_BTST)
- Peripheral Enable 0 Register (BSU8_PEN0)

Table 37-64. Memory layout of Bus Support Unit registers of the HSSPI0 group

Offset	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000004	BSU8_BTST 00000000 00000000 00000000 00000000			
0x00000008 - 0x0000000C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000010	BSU8_PEN0 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			

37.2.8.1 Bus Test Register (BSU8_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 12 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU8_BTST)

Figure 37-57. Bus Test Register (BSU8_BTST)

BSU8_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-65. Bus Test Register (BSU8_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register for HSSPI0 Group This register holds the value of test vector. The contents of this register are used for testing the data line. A value written to BSU8_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.8.2 Peripheral Enable 0 Register (BSU8_PEN0)

The Peripheral Enable 0 Register holds the enable bits for the peripheral 0 (HSSPI0) of the HSSPI0 group. This register should be written only with 32-bit accesses.

Peripheral Enable 0 Register (BSU8_PEN0)

Figure 37-58. Peripheral Enable 0 Register (BSU8_PEN0)

BSU8_PEN0																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENHSSPI0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-66. Peripheral Enable 0 Register (BSU8_PEN0) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENHSSPI0	Peripheral Enable bit for HSSPI0 '0': HSSPI0 is not available '1': HSSPI0 is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.2.9 Registers of Bus Support Unit of the EBI group

Memory layout of Bus Support Unit registers of EBI group

- Bus Test Register (BSU10_BTST)
- Peripheral Enable 0 Register (BSU10_PEN0)

Table 37-67. Memory layout of Bus Support Unit registers of EBI group

Offset	+3	+2	+1	+0
0x00000000	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000004	BSU10_BTST 00000000 00000000 00000000 00000000			
0x00000008	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000000C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000010	BSU10_PEN0 XXXXXXXX XXXXXXXX XXXXXXXX 00000000			

37.2.9.1 Bus Test Register (BSU10_BTST)

The Bus Test Register holds the 32-bit value of the test vector. The data written to this register is rotated left by 3 bits. The value from this register can be read back to test the data lines of the bus to check if it matches with the expected data depending on the data written and the rotate operation.

Bus Test Register (BSU8_BTST)

Figure 37-59. Bus Test Register (BSU10_BTST)

BSU10_BTST																															
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**
BTST[31]	BTST[30]	BTST[29]	BTST[28]	BTST[27]	BTST[26]	BTST[25]	BTST[24]	BTST[23]	BTST[22]	BTST[21]	BTST[20]	BTST[19]	BTST[18]	BTST[17]	BTST[16]	BTST[15]	BTST[14]	BTST[13]	BTST[12]	BTST[11]	BTST[10]	BTST[9]	BTST[8]	BTST[7]	BTST[6]	BTST[5]	BTST[4]	BTST[3]	BTST[2]	BTST[1]	BTST[0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37-68. Bus Test Register (BSU10_BTST) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	BTST	Bus Test Register for EBI Group This register holds the value of test vector. The contents of this register are used for testing the data line. A value written to BSU10_BTST is deemed useful if it causes rotation to take place. If no rotation has taken place, then the value should not be used.

37.2.9.2 Peripheral Enable 0 Register (BSU10_PEN0)

The Peripheral Enable 0 Register holds the enable bits for the peripheral 0 (EBI) of the EBI group. This register should be written only with 32-bit accesses.

Peripheral Enable 0 Register (BSU10_PEN0)

Figure 37-60. Peripheral Enable 0 Register (BSU10_PEN0)

BSU10_PEN0																													
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	ENEBI
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 37-69. Peripheral Enable 0 Register (BSU8_PEN0) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	reserved	-
[7:1]	read0	-
[0]	ENEBI	Peripheral Enable bit for EBI '0': EBI is not available '1': EBI is available This bit is writable only once. Any additional write access to this bit is ignored and results in an error condition.

37.3 Operation of Bus Support Unit

The Bus Support Unit of a particular bus allows to the enabling/disabling of the access to each peripheral on the bus, and it offers a bus test register.

Bus test register

- The Bus test register is a 32-bit register used for bus test purposes
- When the CPU writes to this register, the value which is written is rotated left by a constant value which is BSU specific. For more details refer to the individual BTST registers
- This register is cleared by a hard as well as a soft reset
- Access to this register shall be done only with 32-bit access in BSU3, BSU4, BSU5, BSU6, BSU7, BSU8 and BSU10 and only with 16-bit accesses in BSU0 and BSU1. In case of BSU0 and BSU1, the data written to this register is rotated only after the [31:16] bits of this register are written
- When access is other than 32 bits in AHB and eRBUS or 16 bits in RBUS, error response/BECU error is generated by the slave address decoder

Peripheral enable registers

- Each bit with the peripheral enable registers enables the use of a certain peripheral connected to the bus and the use of resource Input Configuration Registers (RICFG) corresponding to this peripheral.
- These registers are writable only once. Error response/BECU error is generated by the slave address decoder for any additional write to these registers
- These registers are cleared only by hard reset
- PEN bits for peripherals not present in a certain device are always read0, and error response/BECU error will not be generated when these bits are written for the first time
- In the case of BSU0 and BSU1, it is strongly recommended to perform accesses only in 16-bit mode. Error response/BECU error is generated for an access other than 16 bits except when explicitly mentioned in the register description
- In the case of BSU3, BSU4, BSU5, BSU6, BSU8 and BSU10, access to these registers shall be done only with 32-bit access. Error response/BECU error will be generated for any access other than 32 bits

BSU and MediaLb

- If MediaLB is used, the BSU Peripheral Slave Group 4 (peri4) minimum clock value should be 50MHz.
- The system must be able to handle a capacity of 6 MByte/sec on the bus. Therefore, depending on the bus traffic (and access to system RAM by other masters), higher frequencies at PERI4 are recommended.

38. Up/Down Counter

This chapter explains the function and operation of the Up/Down Counter.

38.1 Outline of Up/Down Counter (UDC)

The 32-bit Up/Down Counter (UDC) is triggered by input signals. Specifically, the UDC, running in phase difference count mode, is suitable for counting the encoder pulses of motors and other equipment. When the encoder's phase A, phase B and phase Z output signals are applied, the counter can precisely count the rotation angles or number of revolutions.

Features of Up/Down Counter

The 32-bit Up/Down Counter (UDC) can run in different modes. Counting can be controlled by the encoder outputs AIN, BIN, and ZIN. It can generate an interrupt on compare match, underflow/overflow and count direction change. Features of the UDC are:

- Format: 32-bit length or 2 x 16-bit
- Four types of count mode:
  a. Timer mode
  - Count down with the internal clock

  b. Up/down count mode
  - Counting up is triggered by the AIN pin signal
  - Counting down is triggered by the BIN pin signal

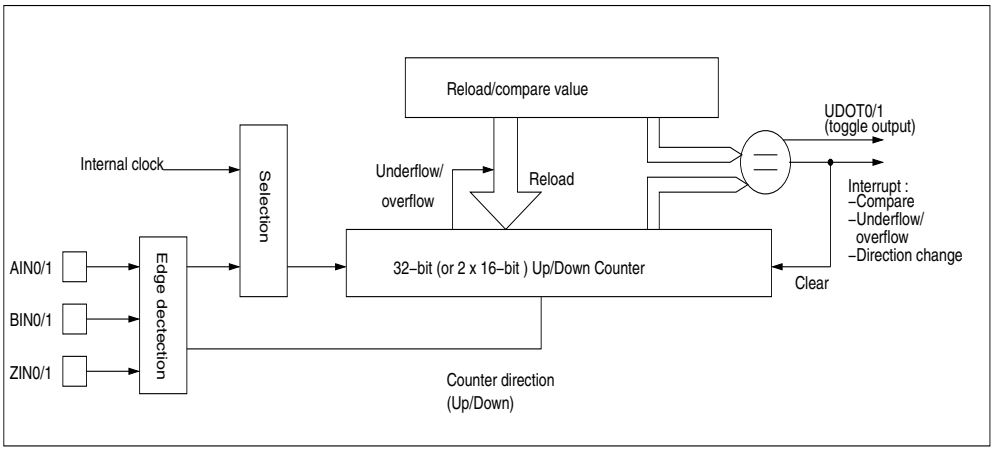
  c. Phase difference count mode (multiply by 2)
  - Counting is triggered by the rising and falling edges of the BIN pin signal. The UDC counts up or down, depending on the AIN pin signal level

  d. Phase difference count mode (multiply by 4)
  - Counting is triggered by the rising and falling edges of the AIN and BIN pin signals
  - If an edge on the AIN pin occurs the counter counts up or down, depending on the BIN pin signal level
  - If an edge on the BIN pin occurs the counter counts up or down, depending on the AIN pin signal level
- Count source
  - Internal clock (timer mode): Peripheral clock (CLK_PERI3_PD2) divided by 2 or 8
  - External trigger (up/down count mode): Edge detection (rising/falling/both edges/no detection)
- Counting range: Any value between 0 and 2^{32}-1 can be set
- Interrupt: Select from the following four types:
  - Compare-match interrupt
  - Underflow interrupt
  - Overflow interrupt
  - Count direction change interrupt
- Others:
  - Whether or not counting is performed can be controlled based on the pin input level
  - The software can activate or deactivate the counter
  - The ZIN pin has two functions: counter clear and gate

- The count direction flag allows identification of the previous count direction
- Toggle output for each channel to keep track of quick changing events
- Support for MCU debug mode

Block diagram of Up/Down Counter

Figure 38-1. Block diagram of Up/Down Counter



Note: In this document, AIN/BIN/ZIN/UDOT can represent AIN0/BIN0/ZIN0/UDOT0 for counter 0 and AIN1/BIN1/ZIN1/ UDOT1 for counter 1.

Configuration diagram of Up/Down Counter

Figure 38-2. Configuration diagram of the Up/Down Counter 0 (16-bit mode)

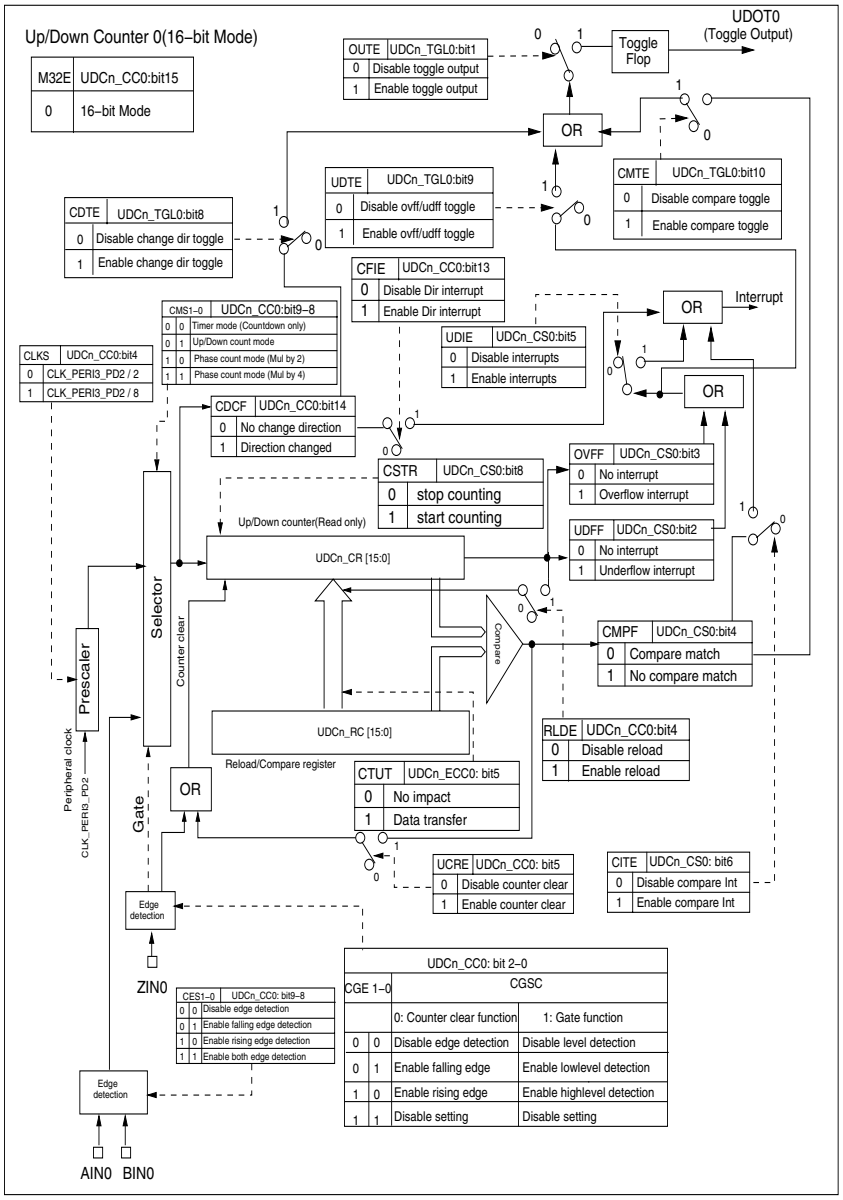


Figure 38-3. Configuration diagram of the Up/Down Counter 1 (16-bit mode)

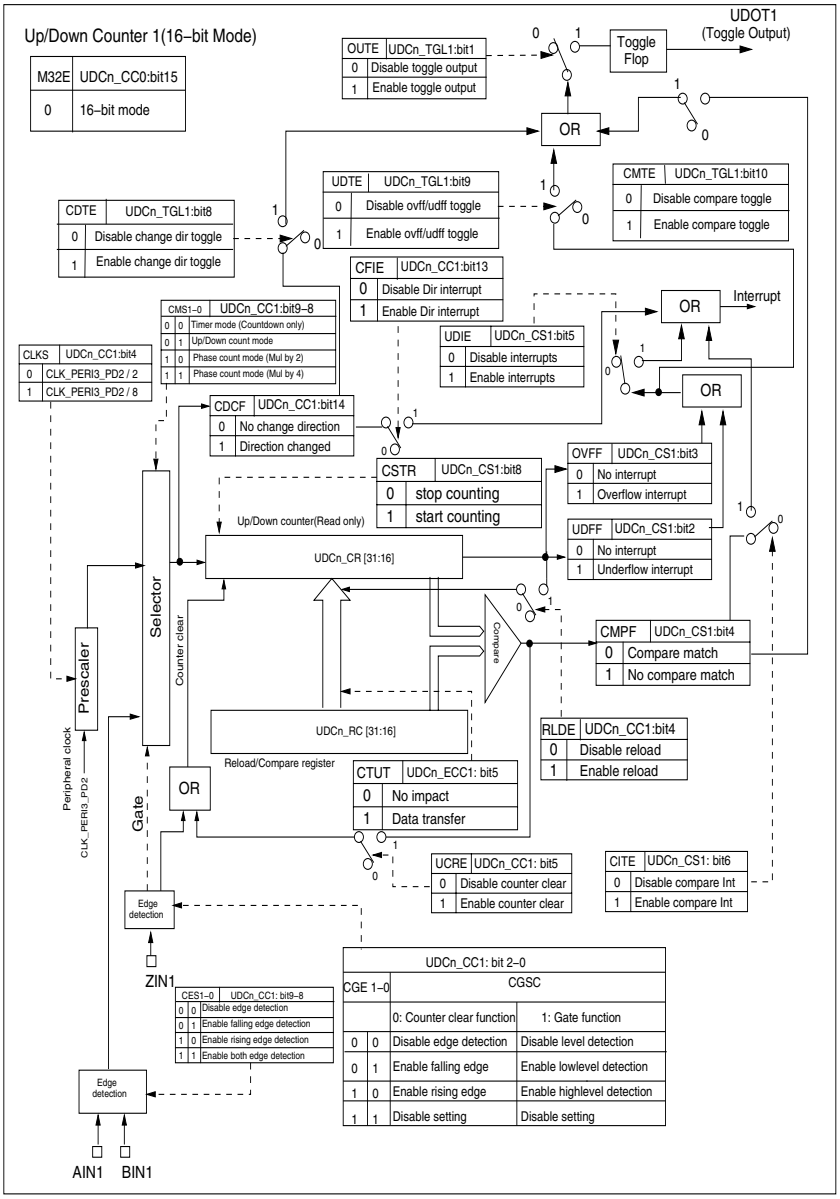
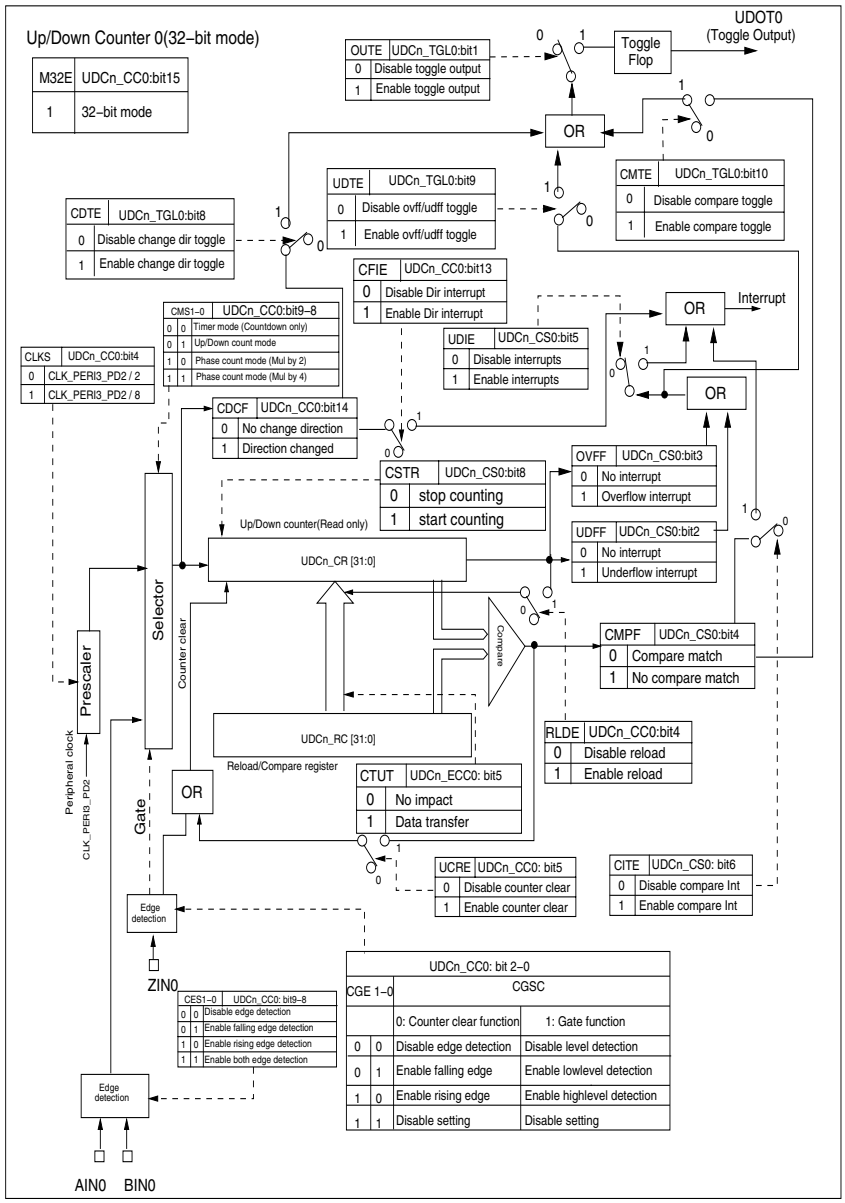


Figure 38-4. Configuration of the Up/Down Counter 0 (32-bit mode)



38.2 Up/Down Counter registers

This section describes the registers of the Up/Down Counter in detail.

The suffix 'n' in the register name indicates that the register is in instance 'n' of the module.

Registers of Up/Down Counter

The following registers are available for each instance of the Up/Down Counter:

- Count Control Register 0 (UDCn_CC0)
- Count Control Register 1 (UDCn_CC1)
- Extended Count Control Register 0 (UDCn_ECC0)
- Extended Count Control Register 1 (UDCn_ECC1)
- Count Status Register 0 (UDCn_CS0)
- Count Status Register 1 (UDCn_CS1)
- Up/Down Counter Register (UDCn_CR)
- Up/Down Reload/Compare Register (UDCn_RC)
- Count Toggle Register 0 (UDCn_TGL0)
- Count Toggle Register 1 (UDCn_TGL1)
- Count Debug Register (UDCn_DBG)

Memory layout of Up/Down Counter registers

Table 38-1. Memory layout of Up/Down Counter registers

Offset	+7	+6	+5	+4	+3	+2	+1	+0
0x00000000	UDCn_ECC1 XXXXXXXX 00000000		UDCn_CC1 00000000 00000000		UDCn_ECC0 XXXXXXXX 00000000		UDCn_CC0 00000000 00000000	
0x00000008	UDCn_TGL1 00000000 00000000		UDCn_TGL0 00000000 00000000		UDCn_CS1 00000000 00000000		UDCn_CS0 00000000 00000000	
0x00000010	UDCn_RC 00000000 00000000 00000000 00000000				UDCn_CR 00000000 00000000 00000000 00000000			
0x00000018	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				reserved XXXXXXXX XXXXXXXX		UDCn_DBG XXXXXXXX 00000000	

38.2.1 Count Control Register 0 (UDCn_CC0)

The Count Control Register 0 controls the operation of the UDC. UDCn_CC0 is used for channel 0 counter control settings when the counter is in 16-bit mode or it is used as a control register for the 32-bit counter when the UDC is configured in 32-bit mode.

Count Control Register 0 (UDCn_CC0)

Figure 38-5. Count Control Register 0 (UDCn_CC0)

UDCn_CC0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
M32E	CDCF	CFIE	CLKS	CMS[1]	CMS[0]	CES[1]	CES[0]	read0	read0	UCRE	RLDE	read0	CGSC	CGE[1]	CGE[0]
RpWp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-2. Count Control Register 0 (UDCn_CC0) bits

Bit Position	Bit Field Name	Bit Description
[15]	M32E	Enable 32-bit Mode '0': UDC works in 16-bit x 2 channel operation mode '1': UDC works in 32-bit x 1 channel operation mode
[14]	CDCF	Count Direction Change Flag Specifies that the direction change has occurred. This flag is cleared by writing '1' to UDCn_CC0:CDCFCLR. '0': No direction change has occurred '1': Direction change has occurred at least once
[13]	CFIE	Count Direction Change Interrupt Enable '0': Disable the Count Direction Change Interrupt Enable '1': Enable the Count Direction Change Interrupt Enable
[12]	CLKS	Select Internal Prescaler This setting is enabled in the timer mode only in which case only countdown is performed. '0': Divide clock by 2 (CLKP/2) '1': Divide clock by 8 (CLKP/8)

Table 38-2. Count Control Register 0 (UDCn_CC0) bits

Bit Position	Bit Field Name	Bit Description
[11:10]	CMS	Select Count Mode These bits are used to select the counter mode. '00': Timer mode (countdown) '01': Up/down count mode '10': Phase difference count mode (multiply by 2) '11': Phase difference count mode (multiply by 4)
[9:8]	CES	Select Count Clock Edge These bita are used in the up/down count mode (UDCn_CC0:CMS = '01') to select the edges to be detected of an AIN and BIN pin signal. This setting is disabled in modes other than the up/down count. '00': Disable edge detection '01': Detect a falling edge '10': Detect a rising edge '11': Detect both rising and falling edges
[7:6]	read0	-
[5]	UCRE	Enable Compare-Match Clear '0': Disable counter clear due to compare match '1': Enable counter clear due to compare match
[4]	RLDE	Enable Reload Reloads up/down counter (UDCn_CR:UDCRL/UDCn_CR) with the reload/compare value (UDCn_RC:UDRCL/UDCn_RC) when the counter is underflowed. '0': Disable reload function '1': Enable reload function
[3]	read0	-
[2]	CGSC	Select ZIN Pin Function '0': ZIN pin works as a counter clear function '1': ZIN pin works as a gate function
[1:0]	CGE	Select Counter Clear/Gate Edge When UDCn_CC0:CGSC = '0', the ZIN0 pin works as a counter clear function with the following functionality defined by these two bits: '00': Disable edge detection '01': Detect a falling edge '10': Detect a rising edge '11': Disable setting When UDCn_CC0:CGSC = '1', the ZIN0 pin works as a gate function with the following functionality: '00': Disable level detection (disable count) '01': Detect an 'L' level '10': Detect an 'H' level '11': Disable setting

38.2.2 Count Control Register 1 (UDCn_CC1)

The Count Control Register 1 controls the operation of the UDC Counter. UDCn_CC1 is used for channel 1 counter control settings when the counter is in 16-bit mode.

Count Control Register 1 (UDCn_CC1)

Figure 38-6. Count Control Register 1 (UDCn_CC1)

UDCn_CC1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	CDCF	CFIE	CLKS	CMS[1]	CMS[0]	CES[1]	CES[0]	read0	read0	UCRE	RLDE	read0	CGSC	CGE[1]	CGE[0]
Rp0	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-3. Count Control Register 1 (UDCn_CC1) bits

Bit Position	Bit Field Name	Bit Description
[15]	read0	-
[14]	CDCF	Count Direction Change Flag Specifies that the direction change has occurred. This flag is cleared by writing '1' to UDCn_CC1:CDCFCLR. '0': No direction change has occurred '1': Direction change has occurred at least once
[13]	CFIE	Count Direction Change Interrupt Enable '0': Disable the Count Direction Change Interrupt Enable '1': Enable the Count Direction Change Interrupt Enable
[12]	CLKS	Select Internal Prescaler This setting is enabled only in the timer mode, in which only count-down is performed. '0': Divide clock by 2 (CLKP/2) '1': Divide clock by 8 (CLKP/8)
[11:10]	CMS	Select Count Mode This bit is used to select the counter mode. '00': Timer mode (countdown) '01': Up/down count mode '10': Phase difference count mode (multiply by 2) '11': Phase difference count mode (multiply by 4)

Table 38-3. Count Control Register 1 (UDCn_CC1) bits

Bit Position	Bit Field Name	Bit Description
[9:8]	CES	Select Count Clock Edge This bit is used in the up/down count mode (UDCn_CC1:CMS = '01') to select the edge, to be detected of an AIN and BIN pin signal. This setting is disabled in modes other than the up/down count '00': Disable edge detection '01': Detect a falling edge '10': Detect a rising edge '11': Detect both rising and falling edges
[7:6]	read0	-
[5]	UCRE	Enable Compare-Match Clear '0': Disable counter clear due to compare match '1': Enable counter clear due to compare match
[4]	RLDE	Enable Reload Reloads up/down counter (UDCn_CR) with reload/compare value (UDCn_RC), when counter is underflowed '0': Disable reload function '1': Enable reload function
[3]	read0	-
[2]	CGSC	Select ZIN Pin Function '0': ZIN pin works as a counter clear function '1': ZIN pin works as a gate function
[1:0]	CGE	Select Counter Clear/Gate Edge When UDCn_CC1:CGSC = '0', this bit represents following functionality for ZIN pin. '00': Disable edge detection '01': Detect a falling edge '10': Detect a rising edge '11': Disable setting When UDCn_CC1:CGSC = '1', this bit represents following functionality for ZIN pin. '00': Disable level detection (disable count) '01': Detect an 'L' level '10': Detect an 'H' level '11': Disable setting

38.2.3 Extended Count Control Register 0 (UDCn_ECC0)

The Extended Count Control Register 0 contains some extra control bits. UDCn_ECC0 is used for channel 0 when the UDC is configured for 16-bit mode or it is used for the 32-bit counter when the UDC is configured for 32-bit mode.

Extended Count Control Register 0 (UDCn_ECC0)

Figure 38-7. Extended Count Control Register 0 (UDCn_ECC0)

UDCn_ECC0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	CTUT	UDCLR	CMPFCLR	OVFFCLR	UDFFCLR	CDCFCLR
-	-	-	-	-	-	-	-	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 38-4. Extended Count Control Register 0 (UDCn_ECC0) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:6]	read0	-
[5]	CTUT	Counter Load '0': No impact on operation '1': Loads the counter with the UDCn_RC:UDRCL/UDCn_RC register. In the case that the counter is counting up, this triggers a compare event which immediately clears the counter if UDCn_CC0~1:UCRE is '1' and restarts the counting from '0'. If UDCn_CC0~1:UCRE is '0', counter is not cleared and continues counting till its maximum value Reading this bit always returns '0'.
[4]	UDCLR	Counter Clear '0': No impact on operation. '1': Clears the counter value to '0' Reading this bit always returns '0'.
[3]	CMPFCLR	Compare Interrupt Clear '0': No impact on operation '1': Clears the Compare Interrupt Flag (UDCn_CS0:CMPF) Reading this bit always returns '0'.

Table 38-4. Extended Count Control Register 0 (UDCn_ECC0) bits

Bit Position	Bit Field Name	Bit Description
[2]	OVFFCLR	Overflow Interrupt Clear '0': No impact on operation '1': Clears the Overflow Interrupt Flag (UDCn_CS0:OVFF) Reading this bit always returns '0'.
[1]	UDFFCLR	Underflow Interrupt Clear '0': No impact on operation '1': Clears the Underflow Interrupt Flag (UDCn_CS0:UDFF) Reading this bit always returns '0'.
[0]	CDCFCLR	Count Direction Interrupt Clear '0': No impact on operation '1': Clears Count Direction Change Interrupt Flag (UDCn_CC0:CDCF) Reading this bit always returns '0'.

38.2.4 Extended Count Control Register 1 (UDCn_ECC1)

The Extended Count Control Register 1 contains some extra control bits. UDCn_ECC1 is used for channel 1 when the UDC is configured for 16-bit mode.

Extended Count Control Register 1 (UDCn_ECC1)

Figure 38-8. Extended Count Control Register 1 (UDCn_ECC1)

UDCn_ECC1															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	CTUT	UDCLR	CMPFCLR	OVFFCLR	UDFFCLR	CDCFCLR
-	-	-	-	-	-	-	-	Rp0	Rp0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 38-5. Extended Count Control Register 1 (UDCn_ECC1) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:6]	read0	-
[5]	CTUT	Counter Load '0': No impact on operation '1': The counter is loaded with UDCn_RC:UDRCL/UDCn_RC register. If the counter is counting up, a compare event is triggered which immediately clears the counter if UDCn_CC0:UCRE is '1' and restarts the count from '0'. If UDCn_CC0:UCRE is '0', the counter is not cleared and continues counting until it reaches its maximum value Reading this bit always returns '0'.
[4]	UDCLR	Counter Clear '0': No impact on operation '1': Clears the counter value to '0' Reading this bit always returns '0'.
[3]	CMPFCLR	Compare Interrupt Clear '0': No impact on operation '1': Clears Compare Interrupt Flag (UDCn_CS1:CMPF) Reading this bit always returns '0'.

Table 38-5. Extended Count Control Register 1 (UDCn_ECC1) bits

Bit Position	Bit Field Name	Bit Description
[2]	OVFFCLR	Overflow Interrupt Clear '0': No impact on operation '1': Clears overflow interrupt flag (UDCn_CS1:OVFF) Reading this bit always returns '0'.
[1]	UDFFCLR	Underflow Interrupt Clear '0': No impact on operation '1': Clears Underflow Interrupt Flag (UDCn_CS1:UDFF) Reading this bit always returns '0'.
[0]	CDCFCLR	Count Direction Interrupt Clear '0': No impact on operation '1': Clears Count Direction Change Interrupt flag (UDCn_CC1:CDCF) Reading this bit always returns '0'.

38.2.5 Count Status Register 0 (UDCn_CS0)

The Count Status Register 0 is used to control the UDC and indicates the status of the counter. UDCn_CS0 is used for channel 0 when the UDC is configured for 16-bit mode or it is used for the 32-bit counter when the UDC is configured for 32-bit mode.

Count Status Register 0 (UDCn_CS0)

Figure 38-9. Count Status Register 0 (UDCn_CS0)

UDCn_CS0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	CSTR	read0	CITE	UDIE	CMPF	OVFF	UDFF	UDF[1]	UDF[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	RpWp	RpWp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-6. Count Status Register 0 (UDCn_CS0) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	CSTR	Count Operation '0': Disable count operation '1': Enable count operation (activate counter)
[7]	read0	-
[6]	CITE	Compare-Match Interrupt Request Enable '0': Disable compare-match interrupt request '1': Enable compare-match interrupt request
[5]	UDIE	Overflow/Underflow Interrupt Request Enable '0': Disable overflow/underflow interrupt request '1': Enable overflow/underflow interrupt request
[4]	CMPF	Compare-Match Detection Flag '0': Comparison results do not match '1': Comparison results match
[3]	OVFF	Overflow Detection Flag '0': No overflow '1': An overflow has occurred
[2]	UDFF	Underflow Detection Flag ' 0': No underflow '1': An underflow has occurred

Table 38-6. Count Status Register 0 (UDCn_CS0) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	UDF	Up/Down Flag These bits indicate the direction status of the previous count operation. '00': No input '01': Count down '10': Count up '11': Count up and down

38.2.6 Count Status Register 1 (UDCn_CS1)

The Count Status Register 1 is used to control the UDC and indicates the status of the counter. UDCn_CS1 is used for channel 1 when the UDC is configured for 16-bit mode.

Count Status Register 1 (UDCn_CS1)

Figure 38-10. Count Status Register 1 (UDCn_CS1)

UDCn_CS1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	read0	read0	CSTR	read0	CITE	UDIE	CMPF	OVFF	UDFF	UDF[1]	UDF[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	RpWp	RpWp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-7. Count Status Register 1 (UDCn_CS1) bits

Bit Position	Bit Field Name	Bit Description
[15:9]	read0	-
[8]	CSTR	Count Operation '0': Disable Count Operation '1': Enable Count Operation (activate counter)
[7]	read0	-
[6]	CITE	Compare-Match Interrupt Request '0': Disable Compare-Match Interrupt Request '1': Enable Compare-Match Interrupt Request
[5]	UDIE	Overflow/Underflow Interrupt Request '0': Disable Overflow/Underflow Interrupt Request '1': Enable Overflow/Underflow Interrupt Request
[4]	CMPF	Compare-Match Detection Flag '0': Comparison results do not match '1': Comparison results match
[3]	OVFF	Overflow Detection Flag '0': No overflow '1': An overflow has occurred
[2]	UDFF	Underflow Detection Flag '0': No underflow '1': An underflow has occurred

Table 38-7. Count Status Register 1 (UDCn_CS1) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	UDF	Up/Down Flag This bit indicates the direction status of the previous count operation. '00': No input '01': Count down '10': Count up '11': Count up and down

38.2.7 Up/Down Counter Register (UDCn_CR)

The Up/Down Counter Register reflects the current running value of the counter.

Up/Down Counter Register (UDCn_CR)

Figure 38-11. Up/Down Counter Register (UDCn_CR)

UDCn_CR																															
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**
UDCRH[15]	UDCRH[14]	UDCRH[13]	UDCRH[12]	UDCRH[11]	UDCRH[10]	UDCRH[9]	UDCRH[8]	UDCRH[7]	UDCRH[6]	UDCRH[5]	UDCRH[4]	UDCRH[3]	UDCRH[2]	UDCRH[1]	UDCRH[0]	UDCRL[15]	UDCRL[14]	UDCRL[13]	UDCRL[12]	UDCRL[11]	UDCRL[10]	UDCRL[9]	UDCRL[8]	UDCRL[7]	UDCRL[6]	UDCRL[5]	UDCRL[4]	UDCRL[3]	UDCRL[2]	UDCRL[1]	UDCRL[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-8. Up/Down Counter Register (UDCn_CR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	UDCRH	Up/Down Counter Register This register contains the running value of the counter. If UDC is configured in 32-bit mode (UDCn_CC0:M32E = '1') then UDCn_CR acts as a 32-bit register. If UDC is configured in 16-bit mode (UDCn_CC0~1:M32E = '0') then UDCn_CR acts as 2 x 16-bit registers with UDCn_CR:UDCRL (i.e. UDCn_CR[15:0]) used for channel 0 counter and UDCn_CR:UDCRH (i.e. UDCn_CR[31:16]) for channel 1 counter.
[15:0]	UDCRL	Up/Down Counter Register This register contains the running value of the counter. If UDC is configured in 32-bit mode (i.e. UDCn_CC0:M32E = '1') then UDCn_CR acts as a 32-bit register. If UDC is configured in 16-bit mode (UDCn_CC0:M32E = '0') then UDCn_CR acts as 2 x 16-bit registers with UDCn_CR:UDCRL (i.e. UDCn_CR[15:0]) used for channel 0 counter and UDCn_CR:UDCRH (i.e. UDCn_CR[31:16]) for channel 1 counter.

38.2.8 Up/Down Reload/Compare Register (UDCn_RC)

The Up/Down Reload/Compare Register is used to reload a value to the UDC and to compare the register value with the counter value. This register is also used to write to the Up/Down Counter.

Up/Down Reload/Compare Register (UDCn_RC)

Figure 38-12. Up/Down Reload/Compare Register (UDCn_RC)

UDCn_RC																															
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**
UDRCH[15]	UDRCH[14]	UDRCH[13]	UDRCH[12]	UDRCH[11]	UDRCH[10]	UDRCH[9]	UDRCH[8]	UDRCH[7]	UDRCH[6]	UDRCH[5]	UDRCH[4]	UDRCH[3]	UDRCH[2]	UDRCH[1]	UDRCH[0]	UDRCL[15]	UDRCL[14]	UDRCL[13]	UDRCL[12]	UDRCL[11]	UDRCL[10]	UDRCL[9]	UDRCL[8]	UDRCL[7]	UDRCL[6]	UDRCL[5]	UDRCL[4]	UDRCL[3]	UDRCL[2]	UDRCL[1]	UDRCL[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-9. Up/Down Reload/Compare Register (UDCn_RC) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	UDRCH	Up/Down Reload/Compare Register This register contains the reload/compare value. If UDC is configured in 32-bit mode (i.e. UDCn_CC0:M32E = '1') then UDCn_RC acts as a 32-bit register. If UDC is configured in 16-bit mode (i.e. UDCn_CC0:M32E = '0'), then UDCn_RC acts as 2 x 16-bit registers with UDCn_RC:UDRCL (i.e. UDCn_RC[15:0]) used for channel 0 counter and UDCn_RC:UDRCH (i.e. UDCn_RC[31:16]) for channel 1 counter.
[15:0]	UDRCL	Up/Down Reload/Compare Register This register contains the reload/compare value. If UDC is configured in 32-bit mode (i.e. UDCn_CC0:M32E = '1') then UDCn_RC acts as a 32-bit register. If UDC is configured in 16-bit mode (i.e. UDCn_CC0:M32E = '0'), then UDCn_RC acts as 2 x 16-bit registers with UDCn_RC:UDRCL (i.e. UDCn_RC[15:0]) used for channel 0 counter and UDCn_RC:UDRCH (i.e. UDCn_RC[31:16]) for channel 1 counter.

Note:

8-bit writes to UDCn_RC are not supported. Any 8-bit write to this register is blocked, the previous value is retained and a Peripheral Protection Unit (PPU) error is generated.

- The reload and compare values are the same under the following conditions:
  - When the UDC counts up, the value in UDCn_RC is used as a compare value

- When Up/Down Counter counts down, an underflow is generated and the value in UDCn_RC is used as a reload value for reloading. (The UDC counts between zero and the reload/compare value)

- Perform the following procedure to write to the UDC:
  - Stop counting
  - Write a value to the reload/compare register
  - Write '1' to the counter load bit (i.e. UDCn_ECC0~1:CTUT)

38.2.9 Count Toggle Register 0 (UDCn_TGL0)

The Count Toggle Counter Register 0 is used to control the toggle output operation. UDCn_TGL0 is used for channel 0 when the UDC is configured in 16-bit mode or it is used by the 32-bit counter when configured for 32-bit mode.

Count Toggle Register 0 (UDCn_TGL0)

Figure 38-13. Count Toggle Register 0 (UDCn_TGL0)

UDCn_TGL0															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	CMTE	UDTE	CDTE	read0	read0	read0	read0	read0	read0	OUTE	OUTL
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-10. Count Toggle Register 0 (UDCn_TGL0) bits

Bit Position	Bit Field Name	Bit Description
[15:11]	read0	-
[10]	CMTE	Compare Toggle Enable '0': Disable toggling of UDOT0 output on compare events '1': Enable toggling of UDOT0 output on compare events
[9]	UDTE	Overflow/Underflow Toggle Enable '0': Disable toggling of UDOT0 output on overflow/underflow events '1': Enable toggling of UDOT0 output on overflow/underflow events
[8]	CDTE	Dir Change Toggle Enable '0': Disable toggling of UDOT0 output on direction change events '1': Enable toggling of UDOT0 output on direction change events
[7:2]	read0	-
[1]	OUTE	Output Toggle Enable '0': Toggle output UDOT is the same as the UDCn_TGL0:OUTL bit value '1': Toggle output is enabled and is sensitive to the compare, overflow/underflow or direction change events, depending on their respective enables
[0]	OUTL	Toggle Load Value If UDCn_TGL0:OUTE is '0', the toggle output UDOT0 and this bit are the same. To drive UDOT0, this bit is loaded with the desired value while UDCn_TGL0:OUTE is '0'.

Note: The initial value of the UDOT0 toggle output is logic '0'.

38.2.10 Count Toggle Register 1 (UDCn_TGL1)

The Count Toggle Counter Register 1 is used to control the toggle output operation. UDCn_TGL1 is used for channel 1 when the UDC is configured in 16-bit mode.

Count Toggle Register 1 (UDCn_TGL1)

Figure 38-14. Count Toggle Register 1 (UDCn_TGL1)

UDCn_TGL1															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	read0	read0	CMTE	UDTE	CDTE	read0	read0	read0	read0	read0	read0	OUTE	OUTL
Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38-11. Count Toggle Register 1 (UDCn_TGL1) bits

Bit Position	Bit Field Name	Bit Description
[15:11]	read0	-
[10]	CMTE	Compare Toggle Enable '0': Disable toggling of UDOT1 output on compare events '1': Enable toggling of UDOT1 output on compare events
[9]	UDTE	Overflow/Underflow Toggle Enable '0': Disable toggling of UDOT1 output on overflow/underflow events '1': Enable toggling of UDOT1 output on overflow/underflow events
[8]	CDTE	Dir Change Toggle Enable '0': Disable toggling of UDOT1 output on direction change events '1': Enable toggling of UDOT1 output on direction change events
[7:2]	read0	-
[1]	OUTE	Output Toggle Enable '0': Toggle output UDOT is same as the UDCn_TGL1:OUTL bit value '1': Toggle output is enabled and is sensitive to the compare, overflow/underflow, or direction change events, depending on their respective enables
[0]	OUTL	Toggle Load Value If UDCn_TGL1:OUTE is '0', the toggle output UDOT1 and this bit are the same. To drive UDOT1, this bit is loaded with the desired value while UDCn_TGL1:OUTE is '0'.

Note: The initial value of the UDOT1 toggle output is logic ‘0’.

38.2.11 Count Debug Register (UDCn_DBG)

The Count Debug Register is used to configure the operation of the UDC in debug mode.

Count Debug Register (UDCn_DBG)

Figure 38-15. Count Debug Register (UDCn_DBG)

UDCn_DBG															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	DBGEN
-	-	-	-	-	-	-	-	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp
X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 38-12. Count Debug Register (UDCn_DBG) bits

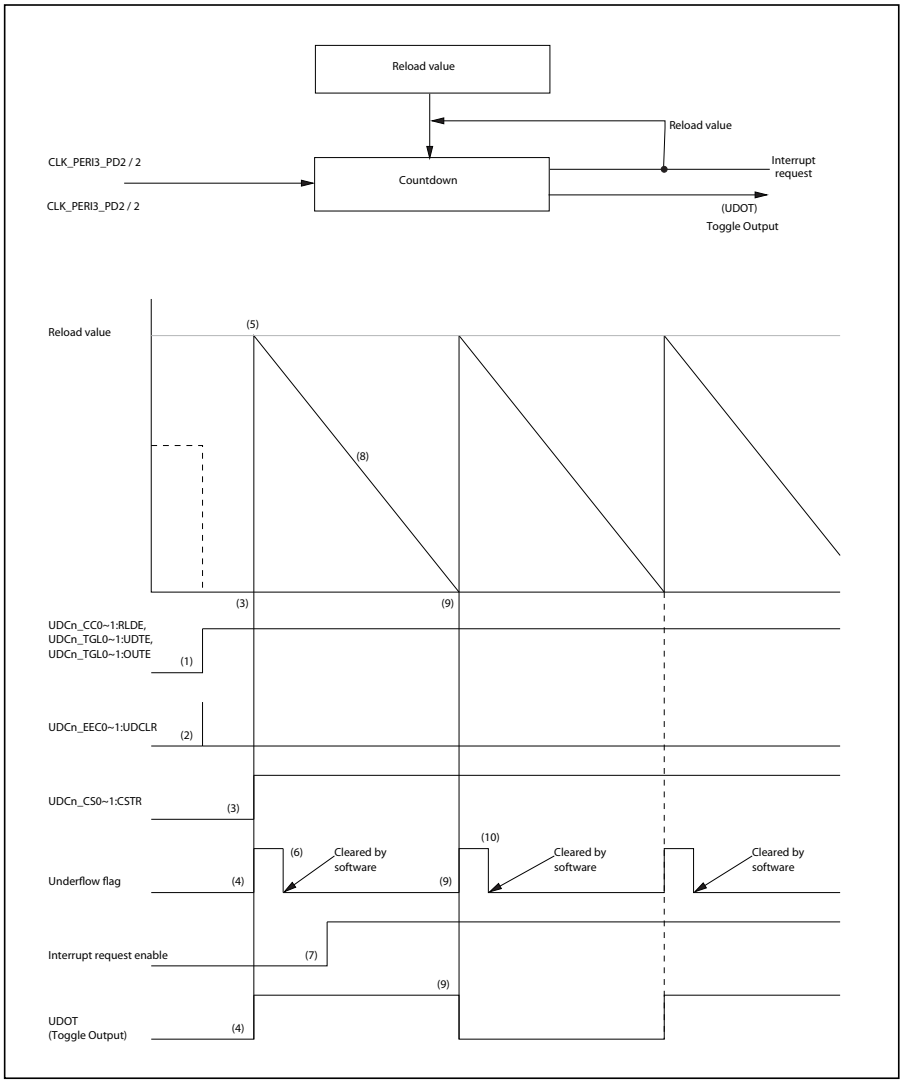
Bit Position	Bit Field Name	Bit Description
[15:8]	reserved	-
[7:1]	read0	-
[0]	DBGEN	Debug Enable '0': Debug mode disabled '1': Debug mode enabled When this bit is '1' and the processor is in Debug state, the counters stop counting and retain the same value until the processor leaves Debug state or DBGEN is set to '0'. For the definition of the debug state, refer to Section 11.8 of the Arm® Cortex®-R4 Technical Reference Manual.

38.3 Operation of Up/Down Counter

This section describes the operation of the UDC in different modes.

Timer mode UDCn_CC0~1:CMS[1:0]= '00'

Figure 38-16. Up/Down Counter in timer mode UDCn_CC0~1:CMS[1:0] = '00'

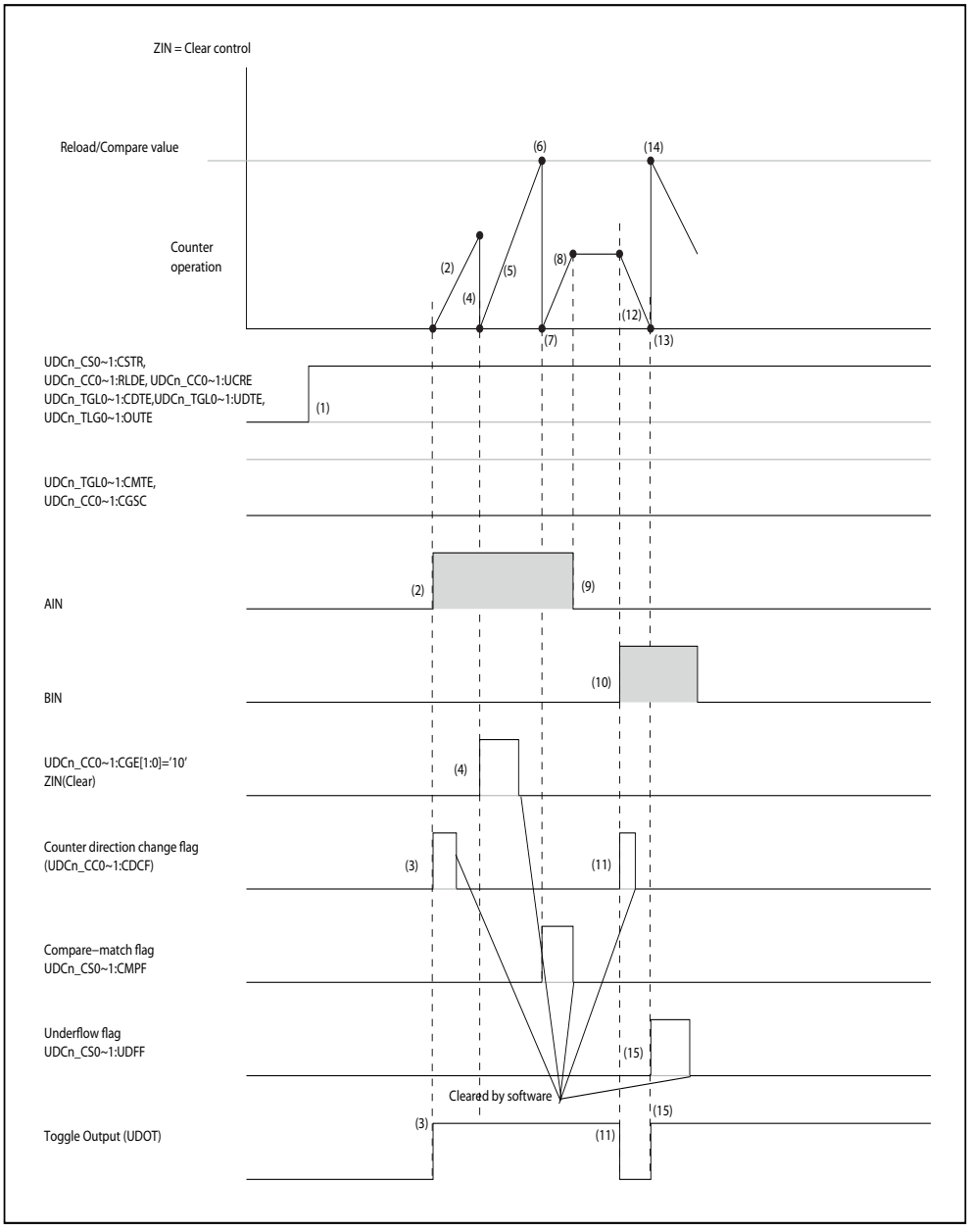


1. An appropriate bit (Reload enable UDCn_CC0~1:RLDE) is set. Toggle Output is configured for underflow events (UDCn_TGL0~1:OUTE = '1' and UDCn_TGL0~1:UDTE = '1').
2. The UDC is cleared (i.e. '1' is written to UDCn_ECC0~1:UDCLR).
3. The software activates the UDC.
4. An underflow occurs. The Toggle Output (UDOT) toggles because of underflow event.
5. The reload value is reloaded to the UDC.
6. The software clears the underflow flag.
7. The software enables the interrupts.
8. The UDC counts down.

9. An underflow occurs. An interrupt request has been made. Also the Toggle Output (UDOT) toggles because of an underflow event.
10. The software clears the underflow flag.
11. Repeat (8) to (10).

Up/Down Count mode UDCn_CC0~1:CMS[1:0] = '01'

Figure 38-17. Up/Down Counter in Up/Down count mode UDCn_CC0~1:CMS[1:0] = ’01'



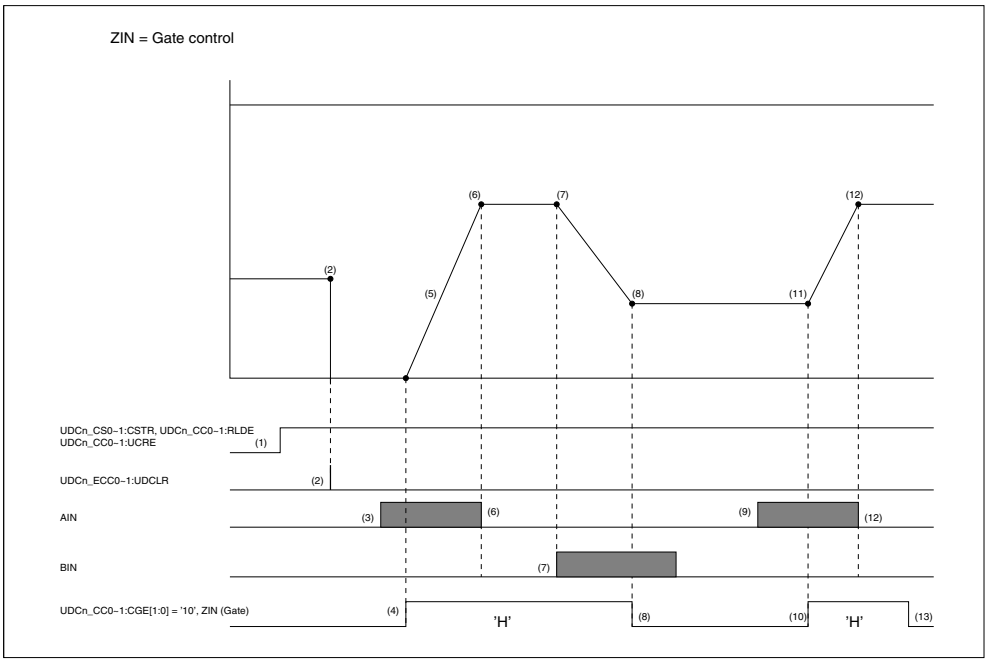
- UDC clear control using the ZIN pin

1. The appropriate bits (counting enable UDCn_CS0~1:CSTR, reload enable UDCn_CC0~1:RLDE and clear enable UDCn_CC0~1:UCRE) are set. Toggle Output is configured by setting the toggle enable UDCn_TGL0~1:OUTE to '1' and becomes sensitive to change direction and underflow/overflow events by enabling UDCn_TGL0~1:CDTE and UDCn_TGL0~1:UDTE respectively.

2. When a pulse input on the AIN pin is detected, the UDC counts up.
3. Toggle Output (UDOT) toggles on a change direction event, and the count direction change flag is set to '1'.
4. When an edge is applied to the ZIN pin, the UDC is cleared.
5. A continuous pulse input on the AIN pin causes the UDC to count up.
6. The UDC's count value matches the compare value (compare-match) and the compare-match flag is set to '1'.
7. The compare-match flag clears the UDC but Toggle Output does not toggle because it is not enabled (UDCn_TGL0~1:CMTE = '0') for compare-match events.
8. A continuous pulse input on the AIN pin causes the UDC to count up.
9. When this pulse input stops, the UDC stops counting.
10. When a pulse input on the BIN pin is detected, the UDC counts down.
11. Toggle Output (UDOT) toggles on a change direction event and the count direction change flag is set to '1'.
12. A continuous pulse input on the BIN pin causes the UDC to count down.
13. The UDC underflows. Toggle Output (UDOT) toggles on an underflow event and the underflow flag is set to '1'.
14. The underflow causes the reload value to be reloaded to UDC.
15. When the UDC next counts down, the compare-match flag is set to '1'.

Up/Down count mode UDCn_CC0~1:CMS[1:0] = '01'

Figure 38-18. Up/Down Counter in Up/Down count mode UDCn_CC0~1:CMS[1:0] = ’01'



1. The appropriate bits (i.e. counting enable UDCn_CS0~1:CSTR, Reload enable UDCn_CC0~1:RLDE and clear enable UDCn_CC0~1:UCRE) are set.
2. The UDC is cleared (i.e. '1' is written to UDCn_ECC0~1:UDCLR).
3. Because a pulse input on the AIN pin and counting on the ZIN pin are not enabled, the UDC neither counts up or down.
4. Counting is enabled on the ZIN pin.
5. The UDC counts up.
6. When the pulse input on the AIN pin stops, the UDC stops counting.
7. When a pulse input is detected on the BIN pin, the UDC counts down.
8. When counting is disabled on the ZIN pin, the UDC stops counting.
9. Because a pulse input on the AIN pin and counting on the ZIN pin are disabled, the UDC neither counts up or down.

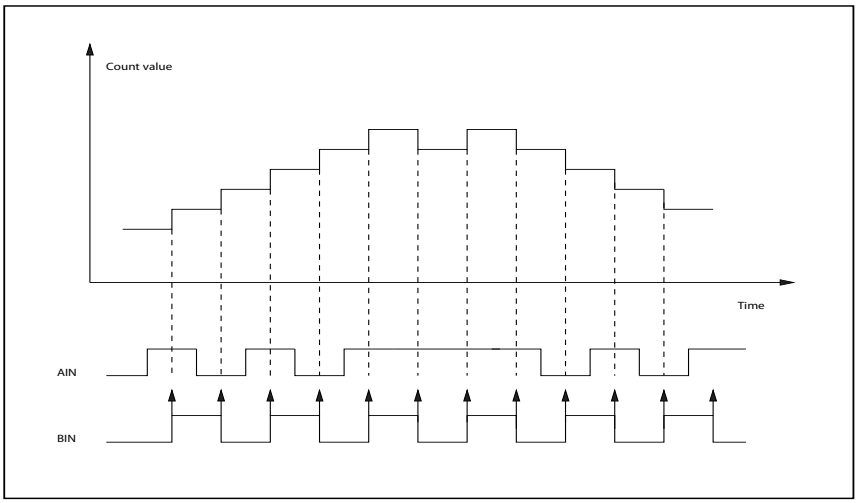
10. Counting is enabled on the ZIN pin.
11. The UDC counts up.
12. When pulse input to the AIN pin stops, Up/Down Counter stops counting.
13. Counting is disabled at the ZIN pin.

Phase difference count mode (multiply by 2) UDCn_CC0~1:CMS[1:0] = '10'

Frequency multiplied by 2 in phase difference count mode:

On the rising and falling edges on the BIN count pin, the UDC counts up or down, depending on the voltage level at the AIN pin.

Figure 38-19. Up/Down Counter in phase difference count mode (multiply by 2) UDCn_CC0~1:CMS[1:0] = '10'



- Count up conditions:
  - When the voltage level on the AIN pin is ‘H’ with a rising edge on the BIN pin
  - When the voltage level on the AIN pin is ‘L’ with a falling edge on the BIN pin
- Count down conditions:
  - When the voltage level on the AIN pin is ‘L’ with a rising edge on the BIN pin
  - When the voltage level at the AIN pin is ‘H’ with a falling edge on the BIN pin

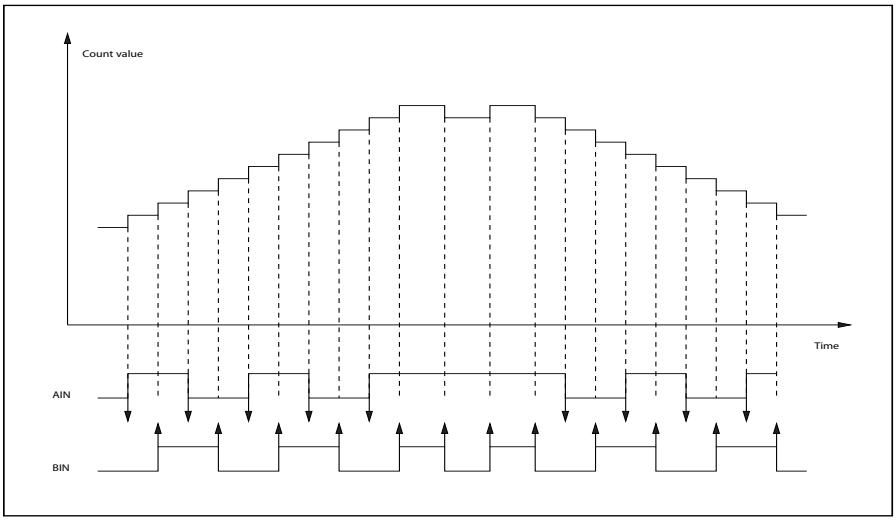
When this count mode is selected, selection of the edge to be detected using UDCn_CC0~1:CES[1:0] is disabled.

Phase difference count mode (multiply by 4) UDCn_CC0~1:CMS[1:0] = '11'

Frequency multiplied by 4 in phase difference count mode:

On the rising and falling edges on the BIN pin, the UDC counts up or down, depending on the voltage level at the AIN pin and on the rising and falling edges at the AIN pin, Up/Down Counter counts up or down, depending on the voltage level at the BIN pin.

Figure 38-20. Up/Down Counter in phase difference count mode (multiply by 4) UDCn_CC0~1:CMS[1:0] = '11'



- Count up conditions:
  - When the voltage level on the AIN pin is ‘H’ with a rising edge on the BIN pin
  - When the voltage level on the AIN pin is ‘L’ with a falling edge on the BIN pin
  - When the voltage level on the BIN pin is ‘L’ with a rising edge on the AIN pin
  - When the voltage level on the BIN pin is ‘H’ with a falling edge on the AIN pin
- Count down conditions:
  - When the voltage level on the AIN pin is ‘L’ with a rising edge on the BIN pin
  - When the voltage level on the AIN pin is ‘H’ with a falling edge on the BIN pin
  - When the voltage level on the BIN pin is ‘H’ with a rising edge on the AIN pin
  - When the voltage level on the BIN pin is ‘L’ with a falling edge on the AIN pin

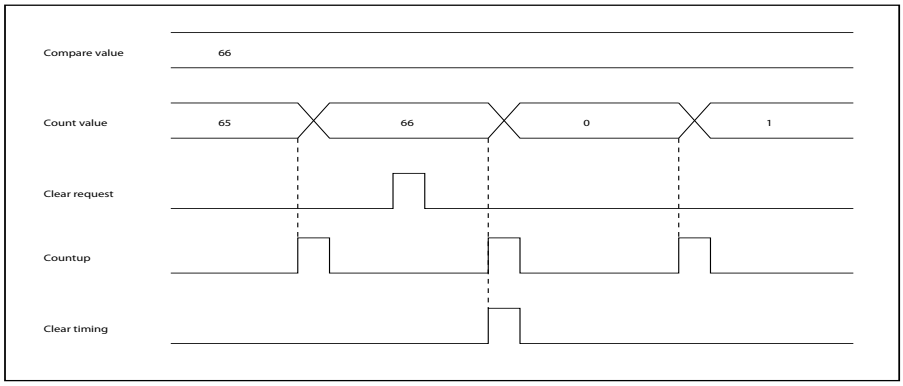
When the UDC is used to count the encoder output, highly precise counting of the rotation angles and number of revolutions, as well the detection of rotation directions, can be achieved by applying phase A and phase B encoder output signals to the AIN and BIN pins respectively.

Note: When this count mode is selected, selection of the edge to be detected using UDCn_CC0~1:CES[1:0] is disabled.

Clear timing for Up/Down Counter

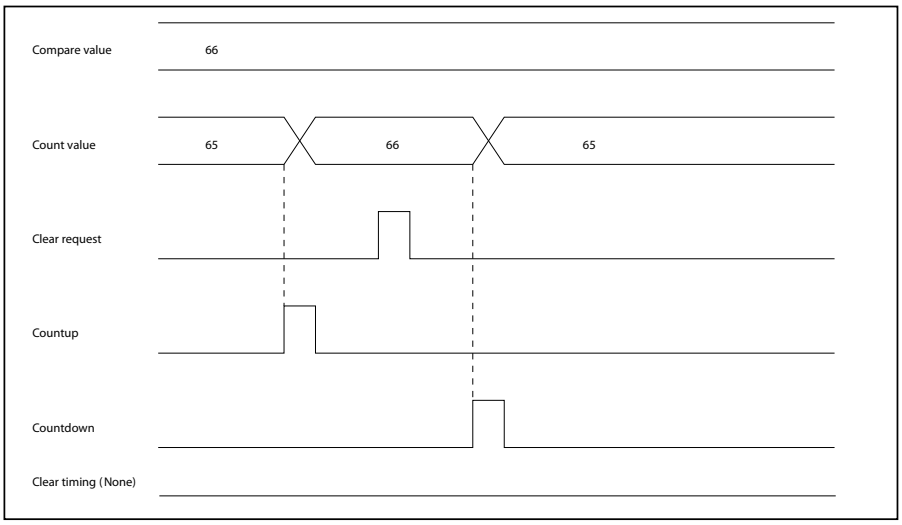
(1) When a clear request (compare-match, ZIN edge detection and writing ’1’ to the clear bit UDCn_ECC0~1:UDCLR) is made, a clear is performed the next time the UDC counts: up in the case of compare-match; up or down in the case of ZIN edge detection; or writing ’1’ to UDCn_ECC0~1:UDCLR.

Figure 38-21. Clear timing of Up/Down Counter (1)



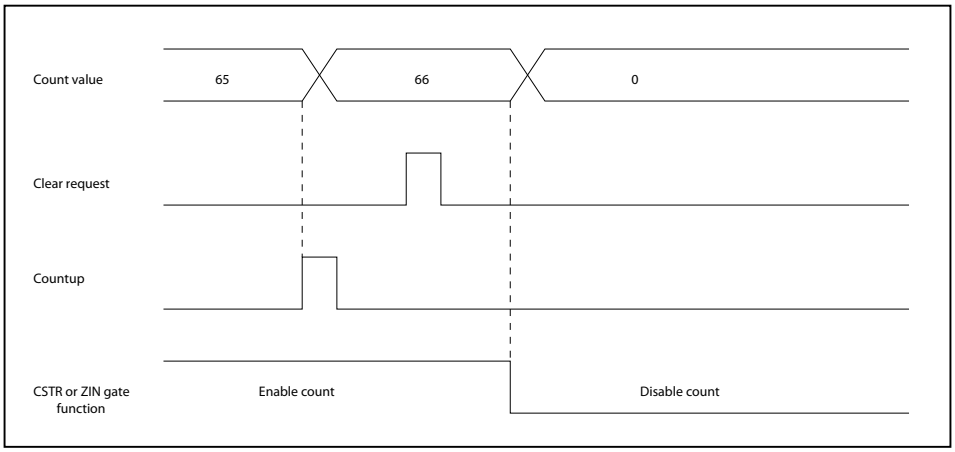
(2) Even if a clear request by compare-match is made, a clear is not performed when the UDC counts down.

Figure 38-22. Clear timing of Up/Down Counter (2)



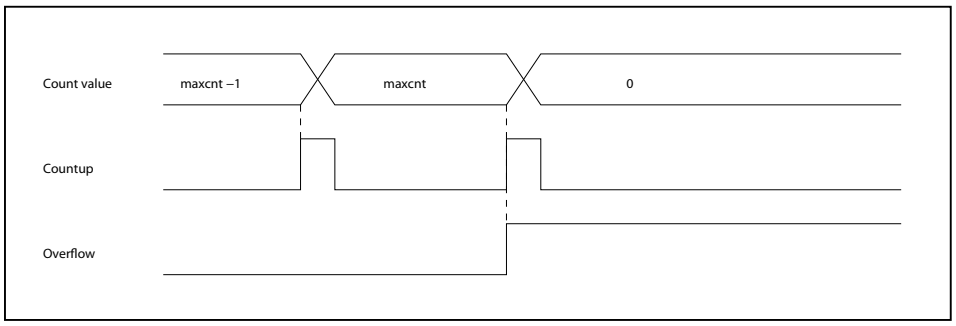
(3) If the UDC does not count up or down after either a clear request by the ZIN edge detection or by writing ’1’ to the clear bit UDCn_ECC0~1:UDCLR is made, the counter is cleared when counting is disabled (i.e. UDCn_CS0~1:CSTR = ’0’).

Figure 38-23. Clear timing of Up/Down Counter (3)



(4) When the UDC exceeds the maximum count, the overflow flag is set to ’1’ and the counter value is returned to zero.

Figure 38-24. Clear timing of Up/Down Counter (4)



Note:

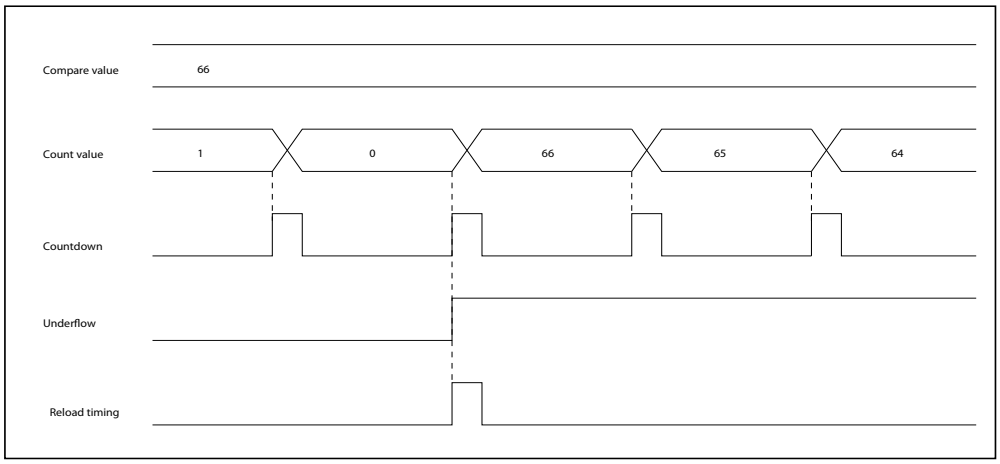
maxcnt = 65535 (0xFFFF) in 16-bit mode (i.e. UDCn_CC0:M32E = ’0’)

maxcnt = 4294967295 (0xFFFFFFFF) in 32-bit mode (i.e. UDCn_CC0:M32E = ’1’)

Reload timing of Up/Down Counter

The next time the UDC counts under zero, an underflow occurs (an interrupt request is generated) and then reloading is performed.

Figure 38-25. Reload timing of Up/Down Counter

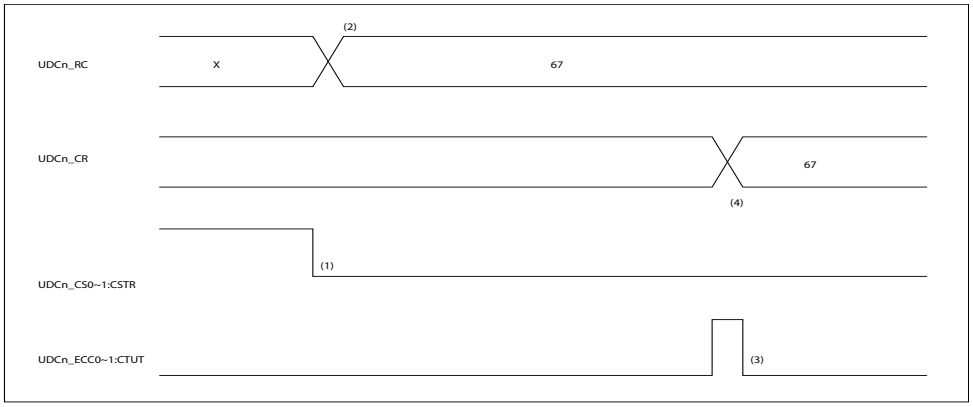


Note:

If clear and reload operations occur at the same time, clear takes precedence.

Writing a value to Up/Down Counter

Figure 38-26. Writing value to Up/Down Counter



1. Counting of UDC is disabled.
2. A value is written to UDCn_RC.
3. '1' is written to the counter load bit UDCn_ECC0~1:CTUT.
4. A value is transferred from the Reload/Compare register UDCn_RC to the UDC.

Counter setting for different modes

Table 38-13. Required settings to run UDC in timer mode

Setting	Setting registers	Setting procedure
(Optional) Set a value to the UDC or Clear the count value of the UDC	Reload/Compare Register (UDCn_RC)	See 38.3.1.5 Setting a value to the UDC
	Extended Count Control Register (UDCn_ECC0~1)	See 38.3.1.8 Method to clear UDC
Set the reload value	Reload/Compare Register (UDCn_RC)	See 38.3.1.16 Method to set the Reload/Compare value

Table 38-13. Required settings to run UDC in timer mode

Setting	Setting registers	Setting procedure
Set a bit length	Count Control Register (UDCn_CC0~1)	See 38.3.1.1 Selection of bit length (32 or 16) of the UDC
Set the count mode to timer mode		See 38.3.1.2 Number of count modes available and how are they set
Select a count source		See 38.3.1.3 Selection of count source for UDC running in the timer mode
Enable reloading at the time of underflow		See 38.3.1.7 Method to reload value (UDCn_RC) to UDC when it has underflowed
Enable count control (clear/gate) using the ZIN pin		See 38.3.1.9 Method to clear UDC using the ZIN pin and 38.3.1.10 Method to control UDC's count operation using the ZIN pin
Activate UDC	Count Status Register (UDCn_CS0~1)	See 38.3.1.11 Method to enable/disable UDC's count operation

Table 38-14. Required settings to run UDC in Up/Down Count mode

Setting	Setting registers	Setting procedure
(Optional) Set a value to UDC Or Clear the count value of UDC	Reload/Compare Register (UDCn_RC)	See 38.3.1.5 Setting a value to the UDC
	Extended Count Control Register (UDCn_ECC0~1)	See 38.3.1.8 Method to clear UDC
Set the reload value/compare value	Reload/Compare Register (UDCn_RC)	See 38.3.1.16 Method to set the Reload/Compare value

Table 38-14. Required settings to run UDC in Up/Down Count mode

Setting	Setting registers	Setting procedure
Set a bit length	Count Control Register (UDCn_CC0~1)	See 38.3.1.1 Selection of bit length (32 or 16) of the UDC
Set the count mode to up/down count mode.		See 38.3.1.2 Number of count modes available and how are they set
Select the edge of a signal (AIN or BIN) for which counting is performed		See 38.3.1.4 Selection of edge with which UDC running in the up/down count mode detects an input signal (AIN or BIN)
Enable clearing of UDC at the time of the counting following a compare-match		See 38.3.1.6 How to enable clearing of the UDC the next time the counter counts up after the UDC's count-up value matches the compare value (UDCn_RC)
Enable reloading at the time of underflow		See 38.3.1.7 Method to reload value (UDCn_RC) to UDC when it has underflowed
Enable count control (clear/gate) using the ZIN pin		See 38.3.1.9 Method to clear UDC using the ZIN pin and 38.3.1.10 Method to control UDC's count operation using the ZIN pin
Activate UDC	Count Status Register (UDCn_CS0~1)	See 38.3.1.11 Method to enable/disable UDC's count operation

Table 38-15. Required settings to run UDC in phase difference count mode (multiply by 2 or 4)

Setting	Setting registers	Setting procedure
Set the reload value/compare value	Reload/Compare Register (UDCn_RC)	See 38.3.1.16 Method to set the Reload/Compare value
(optional) Set a value to UDC or Clear the count value of UDC	Reload/Compare Register (UDCn_RC)	See 38.3.1.5 Setting a value to the UDC
	Extended Count Control Register (UDCn_ECC0~1)	See 38.3.1.8 Method to clear UDC
Set a bit length	Count Control Register (UDCn_CC0~1)	See 38.3.1.1 Selection of bit length (32 or 16) of the UDC
Set the count mode to phase difference count mode (multiply by 2 or 4)		See 38.3.1.2 Number of count modes available and how are they set
Enable clearing of Up/Down Counter at the time of the counting following a compare-match		See 38.3.1.6 How to enable clearing of the UDC the next time the counter counts up after the UDC's count-up value matches the compare value (UDCn_RC)
Enable reloading at the time of underflow		See 38.3.1.7 Method to reload value (UDCn_RC) to UDC when it has underflowed
Enable count control (clear/gate) using the ZIN pin		See 38.3.1.9 Method to clear UDC using the ZIN pin and 38.3.1.10 Method to control UDC's count operation using the ZIN pin
Activate UDC	Count Status Register (UDCn_CS0~1)	See 38.3.1.11 Method to enable/disable UDC's count operation

Table 38-16. Required settings for UDC interrupt

Setting	Setting registers	Setting procedure
Set UDC interrupt vectors and UDC interrupt levels	Refer to 17. Interrupt Controller	See 38.3.1.17 Method to configure interrupt-related registers
Set UDC interrupts Clear interrupts Enable interrupt requests	Count Control Register (UDCn_CC0~1) Extended Count Control Register (UDCn_ECC0~1) Count Status Register (UDCn_CS0~1)	See 38.3.1.19 Method to enable (select), disable, or clear interrupts

Table 38-17. Required settings to Deactivate the UDC

Setting	Setting registers	Setting procedure
Deactivate UDC (controlled through the ZIN pin)	Count Control Register (UDCn_CC0~1)	See 38.3.1.10 Method to control UDC's count operation using the ZIN pin
Deactivate UDC	Count Status Register (UDCn_CS0~1)	See 38.3.1.11 Method to enable/disable UDC's count operation

38.3.1 Procedure for configuring the UDC

38.3.1.1 Selection of bit length (32 or 16) of the UDC

Table 38-18. Use of 32-bit mode enable bit (UDCn_CC0:M32E)

Bit length of UDC	32-bit mode enable bit (UDCn_CC0:M32E)
To set the bit length to 16-bit	Set the bit to '0'
To set the bit length to 32-bit	Set the bit to '1'

38.3.1.2 Number of count modes available and how are they set

Table 38-19. Use the count mode selection bits (UDCn_CC0~1:CMS[1:0]) to set a count mode

Count mode	Count mode selection bits (UDCn_CC0~1:CMS[1:0])
To set the count mode to timer	Set the bits to '00'
To set the count mode to up/down count	Set the bits to '01'
To set the count mode to phase difference count mode (Multiply by 2)	Set the bits to '10'
To set the count mode to phase difference count mode (Multiply by 4)	Set the bits to '11'

38.3.1.3 Selection of count source for UDC running in the timer mode

Table 38-20. Use the internal prescaler select bit (UDCn_CC0~1:CLKS)

Count source for timer mode	Internal prescaler select bit (UDCn_CC0~1:CLKS)
To obtain the CLK_PERI3_PD2 divided by 2	Set the bit to '0'
To obtain the CLK_PERI3_PD2 divided by 8	Set the bit to '1'

38.3.1.4 Selection of edge with which UDC running in the up/down count mode detects an input signal (AIN or BIN)

Table 38-21. Use count clock edge select bits (UDCn_CC0~1:CES[1:0])

Edge to be detected by counter	Count clock edge select bits (UDCn_CC0~1:CES[1:0])
To disable detection	Set the bits to '00'
To enable detection of a falling edge	Set the bits to '01'
To enable detection of a rising edge	Set the bits to '10'
To enable detection of both edges	Set the bits to '11'

38.3.1.5 Setting a value to the UDC

A value can be set to the UDC by writing the value to the Reload/Compare Register (UDCn_RC) and then writing '1' to the counter load bit (UDCn_ECC0~1:CTUT).

38.3.1.6 How to enable clearing of the UDC the next time the counter counts up after the UDC's count-up value matches the compare value (UDCn_RC)

Table 38-22. Use the UDC clear enable bit (UDCn_CC0~1:UCRE)

When the count-up value matches with the compare value and then the UDC counts up:	UDC clears enable bit (UDCn_CC0~1:UCRE)
To disable clearing of Up/Down Counter	Set the bit to '0'
To enable clearing of Up/Down Counter	Set the bit to '1'

38.3.1.7 Method to reload value (UDCn_RC) to UDC when it has underflowed

Table 38-23. Use of reload enable bit (UDCn_CC0~1:RLDE)

When the count-up value matches the compare value:	Reload enable bit (UDCn_CC0~1:RLDE)
To disable reloading of the reload value (UDCn_RC) to the UDC	Set the bit to '0'
To enable reloading of the reload value (UDCn_RC) to the UDC	Set the bit to '1'

38.3.1.8 Method to clear UDC

The UDC can be cleared in any of the following ways:

- Writing '1' to the UDC clear bit (UDCn_ECC0~1:UDCLR)

- Applying an edge to the ZIN pin
- When the compare value matches with the UDC's count-up value
- When the UDC tries to count up after reaching the maximum count
- Any kind of reset

38.3.1.9 Method to clear UDC using the ZIN pin

Table 38-24. Use counter clear gate bit (UDCn_CC0~1:CGSC) and counter clear gate edge select bits (UDCn_CC0~1:CGE[1:0]). These bits are enabled in the up/down count mode

ZIN pin input	Counter clear gate bit (UDCn_CC0~1:CGSC)	Counter clear gate edge select bits (UDCn_CC0~1:CGE[1:0])
To disable edge detection	Set the bit to '0'	Set the bits to '00'
To clear Up/Down Counter on the falling edge	Set the bit to '0'	Set the bits to '01'
To clear Up/Down Counter on the rising edge	Set the bit to '0'	Set the bits to '10'

UDCn_CC0~1:CGE[1:0] = '11' indicates that setting is disabled.

38.3.1.10 Method to control UDC's count operation using the ZIN pin

Table 38-25. Use counter clear gate bit (UDCn_CC0~1:CGSC) and counter clear gate edge select bits (UDCn_CC0~1:CGE[1:0]). These settings are enabled for all the count modes

ZIN pin input	Counter clear gate bit (UDCn_CC0~1:CGSC)	Counter clear gate edge select bits (UDCn_CC0~1:CGE[1:0])
To disable level detection (disable counting)	Set the bit to '1'	Set the bits to '00'
To start counting up or down at the 'L' level To stop counting up or down at the 'H' level	Set the bit to '1'	Set the bits to '01'
To stop counting up or down at the 'L' level To start counting up or down at the 'H' level	Set the bit to '1'	Set the bits to '10'

UDCn_CC0~1:CGE[1:0]= '11' indicates that the setting is disabled.

38.3.1.11 Method to enable/disable UDC's count operation

Table 38-26. Use the count activate bit (UDCn_CS0~1:CSTR)

When the count-up value matches the compare value	Count activate bit (UDCn_CS0~1:CSTR)
To disable Up/Down Counter's count operation	Set the bit to '0'
To enable Up/Down Counter's count operation (to activate count operation)	Set the bit to '1'

How to start counting

Timer mode	Counting starts using the internal clock
Up/down count mode	Counting starts when the edge of an AIN or BIN pin input signal is detected
Phase difference count mode	Counting starts when a phase difference between AIN and BIN pins is detected

Note: The count operation enable level must be detected before the ZIN pin's gate function can be selected.

38.3.1.12 Method to know the previous count direction (the current rotation direction)

Table 38-27. Use the up/down flags (UDCn_CS0~1:UDF[1:0])

Up/down flags (UDCn_CS0~1:UDF[1:0])
'00' indicates that no counting is performed after resetting
'01' indicates that counting down is performed
'10' indicates that counting up is performed
'11' indicates that both counting up and down are performed, resulting in no change in the count value

These flags have nothing to do with interrupts. Use the count direction change flag (UDCn_CC0~1:CDCF) for interrupt processing.

38.3.1.13 Method to know count direction changes

Table 38-28. Use the count direction change flag (UDCn_CC0~1:CDCF)

Count direction change flag (UDCn_CC0~1:CDCF)
'0' indicates that no direction change has been made after clearing the flag
'1' indicates that a direction change has been made once or more after clearing the flag

38.3.1.14 Method to know that a compare-match has occurred

Table 38-29. Use the compare-match detection flag (UDCn_CS0~1:CMPF)

Compare-match detection flag (UDCn_CS0~1:CMPF)
'0' indicates that the UDC's count value does not match the compare value after clearing the flag
'1' indicates that the UDC's count value matches the compare value after clearing the flag

Regardless of counter operations (counting up/down, or a value being set or reloaded), the compare-match detection flag is set to '1' when the count value matches the compare value.

38.3.1.15 Method to know that an overflow or underflow has occurred

Table 38-30. Use the overflow detection flag (UDCn_CS0~1:OVFF) and the underflow detection flag (UDCn_CS0~1:UDFF)

UDCn_CS0~1:OVFF = '1' indicates that the UDC has overflowed
UDCn_CS0~1:UDFF = '1' indicates that the UDC has underflowed

38.3.1.16 Method to set the Reload/Compare value

Set a value to the Reload/Compare Registers (UDCn_RC). (This value is used as a compare or reload value)

38.3.1.17 Method to configure interrupt-related registers

Configure the UDC interrupt vectors and UDC interrupt level settings.

The following interrupt flags are not automatically cleared:

- Count direction change (UDCn_CC0~1:CDCF)
- Compare-match detection (UDCn_CS0~1:CMPF)
- Overflow (UDCn_CS0~1:OVFF)
- Underflow (UDCn_CS0~1:UDFF)

The software must write ’1’ to the corresponding interrupt clear bits before control is returned from interrupt processing:

- Count Direction Interrupt Clear (UDCn_ECC0~1:CDCFCLR)
- Compare Interrupt Clear (UDCn_ECC0~1:CMPFCLR)
- Overflow Interrupt Clear (UDCn_ECC0~1:OVFFCLR)
- Underflow Interrupt Clear (UDCn_ECC0~1:UDFFCLR)

38.3.1.18 Number of interrupts available and how they are selected

There are four causes of interrupt generation:

- Count direction change
- Compare-match
- Overflow
- Underflow

An interrupt request is made by combining (logical OR) these four interrupt causes. Each interrupt cause cannot be isolated.

Use the interrupt request enable bit to enable a desired interrupt request.

38.3.1.19 Method to enable (select), disable, or clear interrupts

Interrupt request enable bits and interrupt flags

To enable (select) interrupts, use the following interrupt request enable bits:

- Count direction change interrupt request enable bits (UDCn_CC0~1:CFIE)
- Compare-match interrupt request enable bits (UDCn_CS0~1:CITE)
- Overflow/underflow interrupt request enable bits (UDCn_CS0~1:UDIE)

Table 38-31. Interrupt request enable bits

Requirement	Interrupt request enable bits (UDCn_CC0~1:CFIE, UDCn_CS0~1:CITE, and UDCn_CS0~1:UDIE)
To disable interrupt requests	Set the bit to '0'
To enable interrupt requests	Set the bit to '1'

To clear interrupts, use the following interrupt clear bits:

- For count direction changes (UDCn_ECC0~1:CDCFCLR)
- For compare-match detection (UDCn_ECC0~1:CMPFCLR)
- For overflow (UDCn_ECC0~1:OVFFCLR)
- For underflow (UDCn_ECC0~1:UDFFCLR)

Table 38-32. Clearing interrupts using the following interrupt clear bits

Requirement	Interrupt clear bits (UDCn_ECC0~1:CDCFCLR, UDCn_ECC0~1:CMPFCLR, UDCn_ECC0~1:OVFFCLR, and UDCn_ECC0~1:UDFFCLR)
To clear interrupts	Write '1'

38.3.1.20 Method to configure toggle output for change direction events

- Initialize the Toggle Output (UDOT) by writing '0' to UDCn_TGL0~1:OUTL while UDCn_TGL0~1:OUTE is set to '0'
- Write '1' to Change Direction Toggle Enable bit (UDCn_TGL0~1:CDTE)
- Write ’1’ to Output Toggle Enable bit (UDCn_TGL0~1:OUTE)

38.3.1.21 Method to clear toggle output

- Disable Toggle Output by writing ’0’ to UDCn_TGL0~1:OUTE
- Write the desired value to UDCn_TGL0~1:OUTL
- Enable the Toggle Output (UDOT) operations by writing ’1’ to UDCn_TGL0~1:OUTE

38.3.1.22 Method to stop counters in debug mode

- Enable the debug mode for UDC by writing ’1’ to UDCn_DBG:DBGEN
- When the processor is in debug state, counters stop counting and retain the same value until the processor leaves debug state or UDCn_DBG:DBGEN is set to ’0’
- If UDCn_DBG:DBGEN is set to 0, then irrespective of processor debug state counters run in normal mode

For the definition of debug state, refer to section 11.8 of the Arm® Cortex® -R4 Technical Reference Manual.

Caution

- The count direction is set to ’countdown’ immediately after resetting the counter. Therefore, when the counter counts up immediately after resetting, the count direction change bit (UDCn_CC0~1:CDCF) is set to ’1’ to indicate a direction change has been made
- When the UDCn_CR register has reached the maximum count, the overflow flag is set to ’1’ and counting continues. This time UDCn_CR is cleared
- For the minimum pulse width length, refer to the device-specific datasheet
- If it is determined that a change has been made to the count direction change interrupt and count direction flags, it must be taken into consideration that when several direction changes have been made continuously within a short period of time, the count direction flag may be returned to the original value, which looks as if no change has been made
- The compare-match detection flag (UDCn_CS0~1:CMPF) is set to ’1’ when the UDC's count value matches the compare value during both counting up and down. This flag is also set to ’1’ when:
  - The reload value is reloaded to the UDC
  - The UDC's count value matches the compare value when Up/Down Counter is activated
- The UDC's count value is cleared by a clear request which is generated:
  - On the edge of a signal input from the ZIN pin
  - By writing ’1’ to the UDC clear bit (UDCn_ECC0~1:UDCLR)
  - When the compare value matches the count value the next time the counter is counting up

  In addition, the count value is also set to zero when the counter counts up from the maximum count value, or by any kind of reset
- When Up/Down Counter clear and reload requests are made at the same time, the clear operation has higher priority
- When Up/Down Counter is counting, writing ’1’ to the UDCn_ECC0~1:CTUT bit loads the counter with the UDCn_RC register. In case that the counter is counting up, this triggers a compare event which immediately clears the counter if UDCn_CC0~1:UCRE is ’1’ and restarts the counting from zero. But if UDCn_CC0~1:UCRE is ’0', counter is not cleared and continues counting till its max value
- The software cannot clear the up/down flags (UDCn_CS0~1:UDF[1:0]) to ’00’. Only reset (initialization) can clear the flags to ’00’
- When the UDC is configured for 32-bit mode (UDCn_CC0:M32E = ’1’), then overflow, underflow and compare flags for channel 1 should be ignored

39. Automotive Remote Handler

This chapter explains the functions and operations of the Automotive Remote Handler (ARH).

APIX® is a registered mark of INOVA Semiconductors GmbH

39.1 Outline of ARH

This section describes the features and the block diagram of the ARH module.

Features of ARH

The ARH module supports two channels. The ARH module provides secure handling and control of remote peripheral hardware. It supports remote event handling to react to interrupts from remote peripherals. The features of the ARH module are listed as follows:

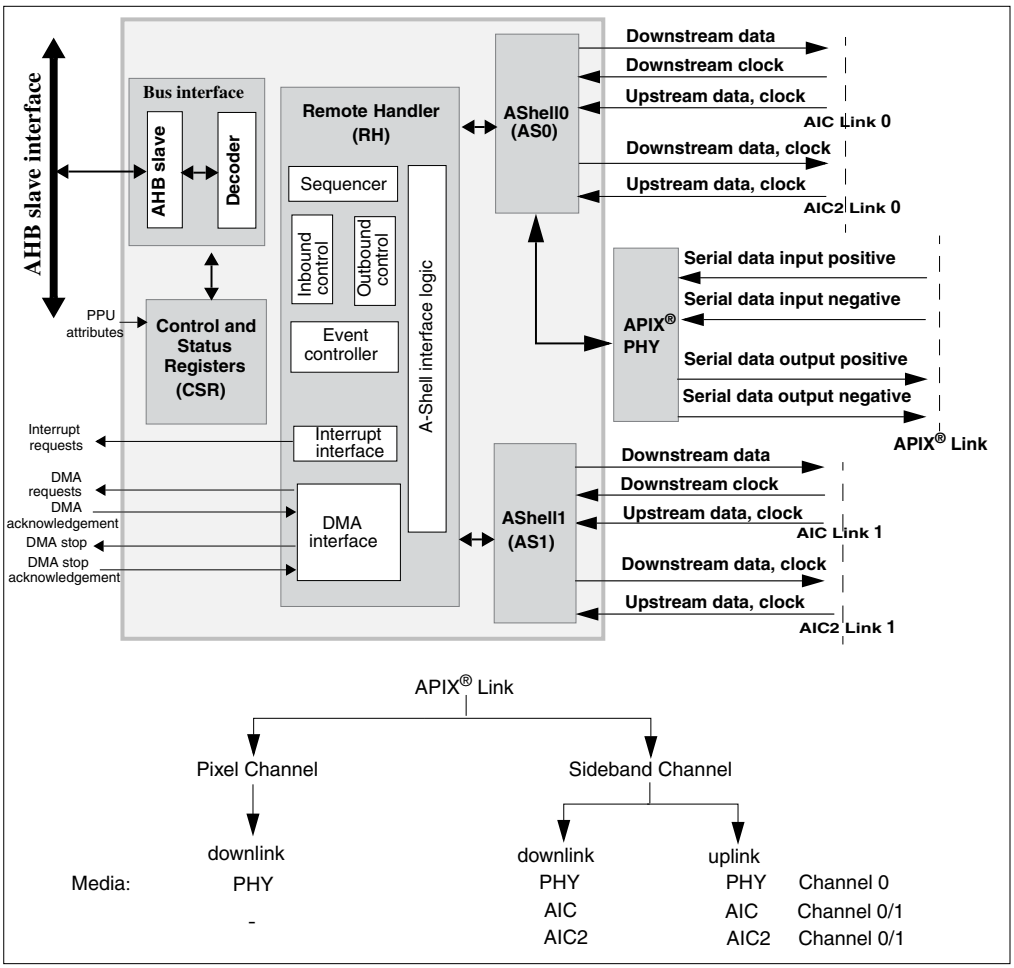
- Supports two Automotive Interconnect links
- Communicates between APIX® PHY and core
- Capable of transmission of write and read messages and reception of read response messages
- Supports up to 16 Transaction Buffers with fixed priority arbitration
- Contains Event Buffer for reception of events from remote device
- Can handle up to 128 events from remote device
- Supports bypass mode through use of Transaction Buffer 0, 1, 2, and 3
- Can handle up to 256 Transaction Frame index
- Supports bulk read and write access (DMA included)
- Support for error handling

Abbreviations

Table 39-1. Abbreviations for ARH

Term	Meaning
AHB	AMBA High Speed Bus
AIC	Automotive interconnect
APIX®	Automotive Pixel Link
A-Shell	Automotive Shell
ARH	Automotive Remote Handler
BSU	Bus Support Unit
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSR	Control and Status Registers
EVBUF	Event Buffer
RH	Remote Handler
TB	Transaction Buffer
TF	Transaction Frame
AAC	A-Shell-to-A-Shell Connector
LPF	Low Pass Filter
RX	Receiver
TX	Transmitter
PRBS	Pseudo Random Binary Sequence
BIST	Built in Self Test
Misc	Miscellaneous

Block diagram of ARHBlock diagram of ARH



- Control and Status Registers

  The operation of ARH can be controlled and monitored through its Control and Status Registers (CSR). For details of the CSR refer to 39.2 ARH registers.

- AHB interface and address decoding

  The AHB masters in the MCU can access the ARH module through its AHB slave interface. The address decoder decodes the AHB address bus. The AHB slave decoder also handles PPU attributes.

- DMA interface

  The ARH supports 16 DMA channels, each per Transaction Buffer. In case of a severe error (fatal) from A-Shell detected, the DMA transfers can be stopped by issuing a DMA stop request.

- Remote Handler

  The ARH Remote Handler block generates and handles various message types. It generates write messages, read request messages, and sends them to A-Shell. This then performs low level services on them and transmits them to the remote peripheral. The A-Shell receives read response messages and event messages from remote peripheral, performs low level services, and forwards the received frame to the Remote Handler for further processing. The Remote Handler on reception of frame from A-Shell, decodes the frame and stores the information for further processing by software.

- A-Shell

  The A-Shell interfaces between the Remote Handler and the remote peripheral. It provides various low level services which include transaction framing and de-framing, establishing and maintaining transaction alignment, exchange of transaction utilization services, bit error detection, bit error management, and report of status information.

Connection over APIX® PHY, AIC or AIC2 link is possible.

Note: Please refer to the device datasheet for the availability of APIX® PHY, AIC or AIC2 link.

39.2 ARH registers

The ARH module contains various registers to configure its operation, to monitor its status and to read the information it has collected from the AHB master interface at the time of the memory protection violation.

The ARH module is allocated 1 KB of MCU address space for mapping the Configuration and Status Registers (i.e. CSRs). This section describes the registers of the ARH in detail.

The suffix 'n' in the register name indicates that the register is an instance 'n' of the module.

Registers of ARH

The following registers are available for ARH:

- ARH Remote Handler Control Register (ARHn_RHCTRL)
- ARH Channel Control Register (ARHn_CHCTRL0 - Channel 0)
- ARH Channel Control Register (ARHn_CHCTRL1 - Channel 1)
- ARH Channel Status Register (ARHn_CHSTAT0 - Channel 0, ARHn_CHSTAT1 - Channel 1)
- ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0 - Channel 0, ARHn_CHWDGCTL1 - Channel 1)
- ARH Watchdog Counter Register (ARHn_CHWDGCNT0 - Channel 0, ARHn_CHWDGCNT1 - Channel 1)
- ARH Transaction Buffer Control Register (ARHn_TBCTRL0~15)
- ARH Transaction Buffer Interrupt Register (ARHn_TBIRQ)
- ARH Transaction Buffer Index Register (ARHn_TBIDX0 - Channel 0, ARHn_TBIDX1 - Channel 1)
- ARH Transaction Frame Control Register (ARHn_TFCTRL00~15)
- ARH Transaction Frame Index Register (ARHn_TFIDX0~15)
- ARH Transaction Frame Address Register (ARHn_TFADDR0~15)
- ARH Transaction Frame Data Register (ARHn_TFDATA0~15)
- ARH Event Control Register (ARHn_EVCTRL)
- ARH Event Interrupt Control Register (ARHn_EVIRQC)
- ARH Event Buffer Register (ARHn_EVBUF0)
- ARH Event Buffer Register (ARHn_EVBUF1)
- ARH APIX® Configuration Register (ARHn_APCFG00 - Channel 0)
- ARH APIX® Configuration Register (ARHn_APCFG10 - Channel 1)
- ARH APIX® Configuration Register (ARHn_APCFG01 - Channel 0)
- ARH APIX® Configuration Register (ARHn_APCFG11 - Channel 1)
- ARH APIX® Configuration Register (ARHn_APCFG02 - Channel 0)
- ARH APIX® Configuration Register (ARHn_APCFG12 - Channel 1)
- ARH APIX® Configuration Register (ARHn_APCFG03 - Channel 0, ARHn_APCFG13 - Channel 1) ARH APIX® Configuration Register (ARHn_APCFG04 - Channel 0, ARHn_APCFG14 - Channel 1)
- ARH Test Register (ARHn_TST)
- ARH Unlock Register (ARHn_UNLOCK)
- ARH Module ID Register (ARHn_MID)
- APIX® PHY Evaluation Transmitter Pattern Register (ARHn_EVAL0)
- APIX® PHY Evaluation Receiver Pattern Register (ARHn_EVAL1)
- APIX® PHY Evaluation Configuration Register (ARHn_EVAL2)
- APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3)
- APIX® PHY Evaluation Misc Test Enable Register (ARHn_EVAL4)
- APIX® PHY Evaluation Status Register (ARHn_EVAL5)

- APIX® PHY Evaluation Misc Test Configuration Register (ARHn_EVAL6)
- APIX® PHY Evaluation Misc Test Status Register (ARHn_EVAL7)

Memory layout of ARH registers

Table 39-2. Memory layout of ARH registers

Offset	+3	+2	+1	+0
0x00000000	ARHn_RHCTRL 00000000 00000000 00000000 00000001			
0x00000004	ARHn_CHCTRL0 00001111 00000000 00000000 00000000			
0x00000008	ARHn_CHSTAT0 00000000 00000000 00000000 00000000			
0x0000000C	ARHn_CHWDGCTL0 00000000 00000000 00000000 00000000			
0x00000010	ARHn_CHWDGCNT0 00000000 00000000 XXXXXXXX XXXXXXXX			
0x00000014	ARHn_CHCTRL1 00000111 00000000 00000000 00000000			
0x00000018	ARHn_CHSTAT1 00000000 00000000 00000000 00000000			
0x0000001C	ARHn_CHWDGCTL1 00000000 00000000 00000000 00000000			
0x00000020	ARHn_CHWDGCNT1 00000000 00000000 XXXXXXXX XXXXXXXX			
0x00000024	ARHn_TBCTRL0 00000000 00000000 00000000 00000000			
0x00000028	ARHn_TBCTRL1 00000000 00000000 00000000 00000000			
0x0000002C	ARHn_TBCTRL2 00000000 00000000 00000000 00000000			
0x00000030	ARHn_TBCTRL3 00000000 00000000 00000000 00000000			
0x00000034	ARHn_TBCTRL4 00000000 00000000 00000000 00000000			
0x00000038	ARHn_TBCTRL5 00000000 00000000 00000000 00000000			
0x0000003C	ARHn_TBCTRL6 00000000 00000000 00000000 00000000			
0x00000040	ARHn_TBCTRL7 00000000 00000000 00000000 00000000			

Table 39-2. Memory layout of ARH registers

Offset	+3	+2	+1	+0
0x00000044	ARHn_TBCTRL8 00000000 00000000 00000000 00000000			
0x00000048	ARHn_TBCTRL9 00000000 00000000 00000000 00000000			
0x0000004C	ARHn_TBCTRL10 00000000 00000000 00000000 00000000			
0x00000050	ARHn_TBCTRL11 00000000 00000000 00000000 00000000			
0x00000054	ARHn_TBCTRL12 00000000 00000000 00000000 00000000			
0x00000058	ARHn_TBCTRL13 00000000 00000000 00000000 00000000			
0x0000005C	ARHn_TBCTRL14 00000000 00000000 00000000 00000000			
0x00000060	ARHn_TBCTRL15 00000000 00000000 00000000 00000000			
0x00000064	ARHn_TBIRQ 00000000 00000000		ARHn_TBIDX0 00000000	ARHn_TBIDX1 00000000
0x00000068	ARHn_TFCTRL0 00000000	ARHn_TFIDX0 00000000	ARHn_TFCTRL1 00000000	ARHn_TFIDX1 00000000
0x0000006C	ARHn_TFCTRL2 00000000	ARHn_TFIDX2 00000000	ARHn_TFCTRL3 00000000	ARHn_TFIDX3 00000000
0x00000070	ARHn_TFCTRL4 00000000	ARHn_TFIDX4 00000000	ARHn_TFCTRL5 00000000	ARHn_TFIDX5 00000000
0x00000074	ARHn_TFCTRL6 00000000	ARHn_TFIDX6 00000000	ARHn_TFCTRL7 00000000	ARHn_TFIDX7 00000000
0x00000078	ARHn_TFCTRL8 00000000	ARHn_TFIDX8 00000000	ARHn_TFCTRL9 00000000	ARHn_TFIDX9 00000000
0x0000007C	ARHn_TFCTRL10 00000000	ARHn_TFIDX10 00000000	ARHn_TFCTRL11 00000000	ARHn_TFIDX11 00000000
0x00000080	ARHn_TFCTRL12 00000000	ARHn_TFIDX12 00000000	ARHn_TFCTRL13 00000000	ARHn_TFIDX13 00000000
0x00000084	ARHn_TFCTRL14 00000000	ARHn_TFIDX14 00000000	ARHn_TFCTRL15 00000000	ARHn_TFIDX15 00000000
0x00000088	ARHn_TFADDR0 00000000 00000000 00000000 00000000			
0x0000008C	ARHn_TFADDR1 00000000 00000000 00000000 00000000			

Table 39-2. Memory layout of ARH registers

Offset	+3	+2	+1	+0
0x00000090	ARHn_TFADDR2 00000000 00000000 00000000 00000000			
0x00000094	ARHn_TFADDR3 00000000 00000000 00000000 00000000			
0x00000098	ARHn_TFADDR4 00000000 00000000 00000000 00000000			
0x0000009C	ARHn_TFADDR5 00000000 00000000 00000000 00000000			
0x000000A0	ARHn_TFADDR6 00000000 00000000 00000000 00000000			
0x000000A4	ARHn_TFADDR7 00000000 00000000 00000000 00000000			
0x000000A8	ARHn_TFADDR8 00000000 00000000 00000000 00000000			
0x000000AC	ARHn_TFADDR9 00000000 00000000 00000000 00000000			
0x000000B0	ARHn_TFADDR10 00000000 00000000 00000000 00000000			
0x000000B4	ARHn_TFADDR11 00000000 00000000 00000000 00000000			
0x000000B8	ARHn_TFADDR12 00000000 00000000 00000000 00000000			
0x000000BC	ARHn_TFADDR13 00000000 00000000 00000000 00000000			
0x000000C0	ARHn_TFADDR14 00000000 00000000 00000000 00000000			
0x000000C4	ARHn_TFADDR15 00000000 00000000 00000000 00000000			
0x000000C8	ARHn_TFDATA0 00000000 00000000 00000000 00000000			
0x000000CC	ARHn_TFDATA1 00000000 00000000 00000000 00000000			
0x000000D0	ARHn_TFDATA2 00000000 00000000 00000000 00000000			
0x000000D4	ARHn_TFDATA3 00000000 00000000 00000000 00000000			
0x000000D8	ARHn_TFDATA4 00000000 00000000 00000000 00000000			

Table 39-2. Memory layout of ARH registers

Offset	+3	+2	+1	+0
0x000000DC	ARHn_TFDATA5 00000000 00000000 00000000 00000000			
0x000000E0	ARHn_TFDATA6 00000000 00000000 00000000 00000000			
0x000000E4	ARHn_TFDATA7 00000000 00000000 00000000 00000000			
0x000000E8	ARHn_TFDATA8 00000000 00000000 00000000 00000000			
0x000000EC	ARHn_TFDATA9 00000000 00000000 00000000 00000000			
0x000000F0	ARHn_TFDATA10 00000000 00000000 00000000 00000000			
0x000000F4	ARHn_TFDATA11 00000000 00000000 00000000 00000000			
0x000000F8	ARHn_TFDATA12 00000000 00000000 00000000 00000000			
0x000000FC	ARHn_TFDATA13 00000000 00000000 00000000 00000000			
0x00000100	ARHn_TFDATA14 00000000 00000000 00000000 00000000			
0x00000104	ARHn_TFDATA15 00000000 00000000 00000000 00000000			
0x00000108	ARHn_EVCTRL 00000000 00000000 00000000 10000000			
0x0000010C	ARHn_EVIRQC 00000000 00000000 00000000 00000000			
0x00000110	ARHn_EVBUF0 0000000X XXXXXXXX 00000000 00000000			
0x00000114	ARHn_EVBUF1 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000118	ARHn_APCFG00 00000000 00110000 00000000 10010000			
0x0000011C	ARHn_APCFG01 11110000 00000000 00000000 01001000			
0x00000120	ARHn_APCFG02 00000010 00000010 01000000 00000000			
0x00000124	ARHn_APCFG03 00100110 10100100 10011010 00000000			

Table 39-2. Memory layout of ARH registers

Offset	+3	+2	+1	+0
0x00000128	ARHn_APCFG10 00000000 00110000 00000000 10010000			
0x0000012C	ARHn_APCFG11 11110000 00000000 00000000 01001000			
0x00000130	ARHn_APCFG12 00000010 00000010 01000000 00000000			
0x00000134	ARHn_APCFG13 00100110 10100100 10011010 00000000			
0x00000138	ARHn_TST 00000000 00000000 00000000 00000000			
0x0000013C	ARHn_UNLOCK 00000000 00000000 00000000 00000000			
0x00000140	ARHn_MID 00000000 00000000 00000000 00000000			
0x00000144	ARHn_APCFG04 00000000 00000000 00000000 00000000			
0x00000148	ARHn_APCFG14 00000000 00000000 00000000 00000000			
0x0000014C	ARHn_EVAL0 00000000 00000000 00000000 00000000			
0x00000150	ARHn_EVAL1 00000000 00000000 00000000 00000000			
0x00000154	ARHn_EVAL2 00000000 00000000 00000000 00001010			
0x00000158	ARHn_EVAL3 00000000 00000000 00000000 00000000			
0x0000015C	ARHn_EVAL4 00000000 00000000 00000000 00001100			
0x00000160	ARHn_EVAL5 00000000 00000000 00000000 00000000			
0x00000164	ARHn_EVAL6 00000000 00000000 00000000 00000000			
0x00000168	ARHn_EVAL7 00000000 00000000 00000000 00000000			

39.2.1 ARH Remote Handler Control Register (ARHn_RHCTRL)

ARH Remote Handler Control Register must be used by the software to program the ARH. This register contains information on ARH module lock/unlock status, it contains various interrupts status information. Trigger to Transaction Buffer, request for cancelling on pending buffer, and request for unlocking on waiting buffer are provided in this register.

ARH Remote Handler Control Register (ARHn_RHCTRL)

Figure 39-1. ARH Remote Handler Control Register (ARHn_RHCTRL)

ARHn_RHCTRL																															
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**
UNLOCK	CANCEL	read0	read0	TBNO[3]	TBNO[2]	TBNO[1]	TBNO[0]	read0	read0	read0	read0	read0	read0	read0	read0	WDG1	WDG0	FAT1	FAT0	read0	LV	OFL	EV	read0	read0	read0	read0	read0	read0	read0	LST
RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp0	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 39-3. ARH Remote Handler Control Register (ARHn_RHCTRL) bits

Bit Position	Bit Field Name	Bit Description
[31]	UNLOCK	Transaction Buffer Unlock Request '0': No unlock request made on TB buffer (ARHn_RHC-TRL:TBNO) which is waiting '1': Request for unlock TB buffer (ARHn_RHCTRL:TBNO) which is in waiting state Caution - Upon unlock request, requested data (response) gets lost after using unlocked TB buffer with the same IDX. For more details refer to Figure 39-9.
[30]	CANCEL	Transaction Buffer Cancel Request '0': No cancel request made on TB buffer (ARHn_RHC-TRL:TBNO) which is in pending or active state '1': Request for cancel TB buffer (ARHn_RHCTRL:TBNO) which is in pending or active state For more details refer to Figure 39-8.
[29:28]	read0	-

Table 39-3. ARH Remote Handler Control Register (ARHn_RHCTRL) bits

Bit Position	Bit Field Name	Bit Description
[27:24]	TBNO	Transaction Buffer Number Trigger Writing starts transaction on buffer number TBNO if ARHn_RHCTRL:UNLOCK and ARHn_RHCTRL:CANCEL bits are '0'. e.g. '0000': Select TBNO 0 for performing transaction ... '1111': Select TBNO 15 for performing transaction
[23:16]	read0	-
[15]	WDG1	Watch Dog Timer 1 Interrupt Read-only interrupt flag indicating the status of channel 1 watchdog transmit or receive interrupts (ARHn_CHWDGCTL1:WDTXIRQx/ ARHn_CHWDGCTL1:WDRXIRQx) depending on interrupt selection (ARHn_CHWDGCTL1:WTTX/ARHn_CHWDGCTL1:WTRX) and when corresponding interrupt enable is high (ARHn_CHWDGCTL1:WDTXIEN/ARHn_CHWDGCTL1:WDRXIEN).
[14]	WDG0	Watch Dog Timer 0 Interrupt Read-only interrupt flag indicating the status of channel 0 watchdog transmit or receive interrupts (ARHn_CHWDGCTL0:WDTXIRQx/ ARHn_CHWDGCTL0:WDRXIRQx) depending on interrupt selection (ARHn_CHWDGCTL0:WTTX/ARHn_CHWDGCTL0:WTRX) and when corresponding interrupt enable is high (ARHn_CHWDGCTL0:WDTXIEN/ARHn_CHWDGCTL0:WDRXIEN).
[13]	FAT1	Fatal Interrupt Status from Channel 1 Read-only flag indicating fatal interrupt status. '0': When ARHn_CHCTRL1:FATIRQ = '1' and ARHn_CHCTRL1:FATEIN = '0' '1': When ARHn_CHCTRL1:FATIRQ = '1' and ARHn_CHCTRL1:FATEIN = '1'
[12]	FAT0	Fatal Interrupt Status from Channel 0 Read-only flag indicating fatal interrupt status. '0': When ARHn_CHCTRL0:FATIRQ = '1' and ARHn_CHCTRL0:FATEIN = '0' '1': When ARHn_CHCTRL0:FATIRQ = '1' and ARHn_CHCTRL0:FATEIN = '1'
[11]	read0	-
[10]	LV	Level Interrupt Status Read-only flag indicating level interrupt status. '0': When ARHn_EVIRQC:EVIRQ = '1' and ARHn_EVIRQC:EVIEN = '0' '1': When ARHn_EVIRQC:EVIRQ = '1' and ARHn_EVIRQC:EVIEN = '1'

Table 39-3. ARH Remote Handler Control Register (ARHn_RHCTRL) bits

Bit Position	Bit Field Name	Bit Description
[9]	OFL	Event Overflow Interrupt Status Read-only flag indicating Event Buffer overflow interrupt status. '0': When ARHn_EVIRQC:OFLIRQ = '1' and ARHn_EVIRQC:OFLIEN = '0' '1': When ARHn_EVIRQC:OFLIRQ = '1' and ARHn_EVIRQC:OFLIEN = '1'
[8]	EV	Event Interrupt Status Read-only flag indicating Event Buffer interrupt status. '0': When ARHn_EVIRQC:EVIRQ = '1' and ARHn_EVIRQC:EVIEN = '0' '1': When ARHn_EVIRQC:EVIRQ = '1' and ARHn_EVIRQC:EVIEN = '1'
[7:1]	read0	-
[0]	LST	ARH Lock Status '0': ARH module is unlocked '1': ARH module is locked **Note:** Following registers of ARH module are affected by LST bit. Registers ARHn_CHCTRL0, ARHn_CHCTRL1, and ARHn_EVCTRL cannot be written when LST = '1'. These registers can be written only when LST = '0'.

39.2.2 ARH Channel Control Register (ARHn_CHCTRL0 - Channel 0)

ARH Channel Control Register can be used by software to program the channel. ARHn_CHCTRL0 register is used for channel 0. This register can be written when ARHn_RHCTRL:LST = '0'.

ARH Channel Control Register (ARHn_CHCTRL0)

Figure 39-2. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0

ARHn_CHCTRL0																															
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**
BYPASS	FATEIN	read0	read0	PHYPLLRST	TXCFG	RSTRTA	INITRH	read0	PHYPLLGOOD	PLLGOOD	UPHSK	DNHSK	FATAL	UPRDY	CONNECTED	UPVALID	DNVALID	FATIRQ	CRCERR	CRCTOUT	PERROR	READY	REMOTERST	UPVALIDCL	DNVALIDST	FATIRQCL	CRCERRCL	CRCTOUTCL	PERRORCL	READYCL	REMOTERSTCL
RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-4. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31]	BYPASS	BYPASS Mode '0': Remote Handler active '1': Remote Handler inactive In bypass mode Transaction Buffer 0 (for A-Shell 0) is used for downstream data (outbound) and Transaction Buffer 1 (for A-Shell 0) is used for upstream data (inbound). Valid written data in Transaction Buffer 0 or 2 is delivered to A-Shell by setting DNVALID. Valid received data in Transaction Buffer 1 or 3 from A-Shell is marked by setting UPVALID.
[30]	FATEIN	FATAL Interrupt Enable '0': Disables FATAL interrupt '1': Enables FATAL interrupt
[29:28]	read0	-
[27]	PHYPLLRST	Reset for PLL '0': Normal Operation '1': Reset Active,

Table 39-4. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[26]	TXCFG	A-Shell Configuration '0': A-Shell and PHY running (write on ARHn_APCFG registers is disabled) '1': Enables A-Shell and PHY configuration (write on ARHn_APCFG registers is enabled which in turn configures A- Shell or PHY)
[25]	RSTRTA	A-Shell Restart '0': A-Shell running '1': A-Shell initialization A-Shell stops operation, clears all state and history information then restart only when RSTRTA = '0'.
[24]	INITRH	Remote Handler Initialization '0': Remote Handler running '1': Remote Handler initialization (no change of TB* and TF* registers) **Note:** ARHn_TBCTRL00:PENDING requests (set while INITRH == 1) is started with INITRH == 0.
[23]	read0	-
[22]	PHYPLLGOOD	APIX® PHY PLL Lock Status '0': APIX® PHY PLL is unlocked '1': APIX® PHY PLL is locked This signal is directly fed into this register bit without any synchronization, indicating the actual status. Remark:For devices which do not contain internal APIX® PHY then this bit is always be read 1. For APIX® PHY availability refer to device specific datasheet.
[21]	PLLGOOD	APIX® PHY PLL Lock Status '0': APIX® PHY PLL is unlocked '1': APIX® PHY PLL is locked This signal is synchronized to reference clock and then reflected in this register bit. When synchronization clock is not available, this bit is not updated. **Note:** For devices which do not contain internal APIX® PHY then this bit is always be read 1. For APIX® PHY availability refer to device specific datasheet.

Table 39-4. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[20]	UPHSK	Upstream Handshake '0': Indicates inbound handshake is in progress, or could not be performed '1': Indicates inbound handshake is performed Local A-Shell performs handshake procedure on request of remote A- Shell.
[19]	DNHSK	Downstream Handshake '0': Indicates outbound handshake is in progress, or could not be performed '1': Indicates outbound handshake is performed Local A-Shell requests and performs handshake procedure with remote A-Shell
[18]	FATAL	Fatal from A-Shell '0': Indicates that A-Shell is running without any severe errors '1': Indicates that A-Shell has encountered conditions where A- Shell cannot continue to deliver and receive transactions. FATAL is only one clock cycle active.
[17]	UPRDY	Upstream Serial Channel Operational '0': Indicates that upstream serial channel (APIX® PHY) is not operational '1': Indicates that upstream serial channel (APIX® PHY) is operational
[16]	CONNECTED	Connection Established '0': Indicates connection to remote APIX® is yet to be established '1': Indicates connection to remote APIX® is established

Table 39-4. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[15]	UPVALID	Upstream Data Valid Status '0': Cleared by SW by setting ARHn_CHCTRL0:UPVALIDCL '1': Set by HW to mark upstream data as valid **Note:** The bypass mode allows direct access and actions on downstream and upstream data and enable ports of the A- Shell (without having the Remote Handler sequencer involved). In bypass mode Transaction Buffer 0 (for A-Shell 0) is used for downstream data (outbound) and Transaction Buffer 1 (for A-Shell 0) is used for upstream data (inbound). In bypass mode Transaction Buffer 2 (for A-Shell 1) is used for downstream data (outbound) and Transaction Buffer 3 (for A-Shell 1) is used for upstream data (inbound). Valid received data in Transaction Buffer 1 or 3 from A-Shell is marked by setting UPVALID.
[14]	DNVALID	Downstream Data Valid Status '0': Cleared by HW after successful transfer to A-Shell '1': Set by SW by writing '1' to ARHn_CHCTRL0:DNVALIDST to mark downstream data as valid **Note:** The bypass mode allows direct access and actions on downstream and upstream data and enable ports of the A- Shell (without having the Remote Handler sequencer involved). In bypass mode Transaction Buffer 0 (for A-Shell 0) is used for downstream data (outbound) and Transaction Buffer 1 (for A-Shell 0) is used for upstream data (inbound). In bypass mode Transaction Buffer 2 (for A-Shell 1) is used for downstream data (outbound) and Transaction Buffer 3 (for A-Shell 1) is used for upstream data (inbound). Valid written data in Transaction Buffer 0 or 2 is delivered to A-Shell by setting DNVALID.
[13]	FATIRQ	Fatal Interrupt Status '0': Fatal interrupt not active '1': Fatal interrupt active, triggered by ARHn_CHCTRL0:FATAL
[12]	CRCERR	CRC Error Status in Upstream Data '0': No CRC error in upstream data (inbound) '1': Indicates occurrence of CRC error in upstream data (inbound)

Table 39-4. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[11]	CRCTOUT	CRC Timeout Status in Upstream Data '0': No occurrence of CRC timeout in upstream data (inbound) '1': Indicates occurrence of CRC timeout in upstream data (inbound)
[10]	PERROR	Protocol Error Status '0': No occurrence of protocol error '1': Indicates occurrence of protocol error
[9]	READY	A-Shell Ready Status '0': Indicates that A-Shell is not ready to accept outbound transactions '1': Indicates that A-Shell is ready to accept outbound transactions
[8]	REMOTERST	Remote A-Shell Restart Status '0': Indicates no restart (transaction realignment procedure) of remote A-Shell is invoked '1': Indicates a restart of remote A-Shell was performed
[7]	UPVALIDCL	Clear Upstream Data Valid Status '0': No effect. Read to this register always reads '0' '1': Clears the upstream data valid status
[6]	DNVALIDST	Set Downstream Data Valid Status '0': No effect. Read to this register always reads '0' '1': Sets the downstream data valid status
[5]	FATIRQCL	Clear Fatal IRQ Interrupt Status '0': No effect. Read to this register always reads '0' '1': Clears the fatal IRQ interrupt status bit ARHn_CHCTRL0:FAT- IRQ
[4]	CRCERRCL	Clear CRC Error in Upstream Data Status '0': No effect. Read to this register always reads '0' '1': Clears the CRC error in upstream data (inbound) status bit ARHn_CHCTRL0:CRCERR
[3]	CRCTOUTCL	Clear CRC Timeout in Upstream Data Status '0': No effect. Read to this register always reads '0' '1': Clears the CRC timeout in upstream data (inbound) status bit ARHn_CHCTRL0:CRCTOUT
[2]	PERRORCL	Clear Protocol Error Status '0': No effect. Read to this register always reads '0' '1': Clears the protocol error status bit ARHn_CHCTRL0:PERROR

Table 39-4. ARH Channel Control Register (ARHn_CHCTRL0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[1]	READYCL	Clear A-Shell Ready Status '0': No effect. Read to this register always reads '0' '1': Clears the A-Shell ready status bit ARHn_CHCTRL0:READY
[0]	REMOTERSTCL	Clear Remote A-Shell Restart Status '0': No effect. Read to this register always reads '0' '1': Clears the remote A-Shell restart status bit ARHn_CHCTRL0:REMOTERST

39.2.3 ARH Channel Control Register (ARHn_CHCTRL1 - Channel 1)

ARH Channel Control Register can be used by software to program the channel. ARHn_CHCTRL1 register is used for channel 1. This register can be written when ARHn_RHCTRL:LST = '0'.

ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

Figure 39-3. ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

ARHn_CHCTRL1																															
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**
BYPASS	FATEIN	read0	read0	read0	TXCFG	RSTRTA	INITRH	read0	read0	PLLGOOD	UPHSK	DNHSK	FATAL	UPRDY	CONNECTED	UPVALID	DNVALID	FATIRQ	CRCERR	CRCTOUT	PERROR	READY	REMOTERST	UPVALIDCL	DNVALIDST	FATIRQCL	CRCERRCL	CRCTOUTCL	PERRORCL	READYCL	REMOTERSTCL
RpWp	RpWp	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-5. ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

Bit Position	Bit Field Name	Bit Description
[31]	BYPASS	BYPASS Mode '0': Remote Handler active '1': Remote Handler inactive In bypass mode Transaction Buffer 0 (for A-Shell 0) is used for downstream data (outbound) and Transaction Buffer 1 (for A-Shell 0) is used for upstream data (inbound). In bypass mode Transaction Buffer 2 (for A-Shell 1) is used for downstream data (outbound) and Transaction Buffer 3 (for A-Shell 1) is used for upstream data (inbound). Valid written data in Transaction Buffer 0 or 2 is delivered to A-Shell by setting ARHn_CHCTRL1:DNVALID. Valid received data in Transaction Buffer 1 or 3 from A-Shell is marked by setting ARHn_CHCTRL1:UPVALID.
[30]	FATEIN	Fatal Interrupt Enable '0': Disables FATAL interrupt '1': Enables FATAL interrupt
[29:27]	read0	-

Table 39-5. ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

Bit Position	Bit Field Name	Bit Description
[26]	TXCFG	A-Shell Configuration '0': A-Shell and PHY running (write on ARHn_APCFG registers is disabled) '1': Enables A-Shell and PHY configuration (write on ARHn_APCFG registers is enabled which in turn configures A- Shell or PHY)
[25]	RSTRTA	A-Shell Restart '0': A-Shell running '1': A-Shell initialization A-Shell stops operation, clears all state and history information then restart only when RSTRTA = '0'.
[24]	INITRH	Remote Handler Initialization '0': Remote Handler running '1': Remote Handler initialization (no change of TB* and TF* registers) **Note:** ARHn_TBCTRL00:PENDING requests (set while INITRH == '1') is started with INITRH == '0'.
[23:22]	read0	-
[21]	PLLGOOD	APIX® PHY PLL Lock Status '0': APIX® PHY PLL is unlocked '1': APIX® PHY PLL is locked This signal is synchronized to reference clock and then reflected in this register bit. When synchronization clock is not available, this bit is not updated. **Note:** For devices which do not contain internal APIX® PHY then this bit is always be read 1. For APIX® PHY availability refer to device specific datasheet.
[20]	UPHSK	Upstream Handshake '0': Indicates inbound handshake is in progress, or could not be performed '1': Indicates inbound handshake is performed Local A-Shell performs handshake procedure on request of remote A- Shell.
[19]	DNHSK	Downstream Handshake '0': Indicates outbound handshake is in progress, or could not be performed '1': Indicates outbound handshake is performed Local A-Shell requests and performs handshake procedure with remote A-Shell

Table 39-5. ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

Bit Position	Bit Field Name	Bit Description
[18]	FATAL	Fatal from A-Shell '0': Indicates that A-Shell is running without any severe errors '1': Indicates that A-Shell has encountered conditions where A- Shell can not continue to deliver and receive transactions. FATAL is only one clock cycle active.
[17]	UPRDY	Upstream Serial Channel Operational '0': Indicates that upstream serial channel (APIX® PHY) is not operational '1': Indicates that upstream serial channel (APIX® PHY) is operational
[16]	CONNECTED	Connection Established '0': Indicates connection to remote APIX® is yet to be established '1': Indicates connection to remote APIX® is established
[15]	UPVALID	Upstream Data Valid Status '0': Cleared by SW by setting ARHn_CHCTRL1:UPVALIDCL '1': Set by HW to mark upstream data as valid **Note:** The bypass mode allows direct access and actions on downstream and upstream data and enable ports of the A- Shell (without having the Remote Handler Sequencer involved). In bypass mode Transaction Buffer 0 (for A-Shell 0) is used for downstream data (outbound) and Transaction Buffer 1 (for A-Shell 0) is used for upstream data (inbound). In bypass mode Transaction Buffer 2 (for A-Shell 1) is used for downstream data (outbound) and Transaction Buffer 3 (for A-Shell 1) is used for upstream data (inbound). Valid received data in Transaction Buffer 1 or 3 from A-Shell is marked by setting UPVALID.

Table 39-5. ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

Bit Position	Bit Field Name	Bit Description
[14]	DNVALID	Downstream Data Valid Status '0': Cleared by HW after successful transfer to A-Shell '1': Set by SW by writing '1'to ARHn_CHCTRL1:DNVALIDST to mark downstream data as valid **Note:** The bypass mode allows direct access and actions on downstream and upstream data and enable ports of the A- Shell (without having the Remote Handler Sequencer involved). In bypass mode Transaction Buffer 0 (for A-Shell 0) is used for down-stream data (outbound) and Transaction Buffer 1 (for A-Shell 0) is used for upstream data (inbound). In bypass mode Transaction Buffer 2 (for A-Shell 1) is used for down-stream data (outbound) and Transaction Buffer 3 (for A-Shell 1) is used for upstream data (inbound). Valid written data in Transaction Buffer 0 or 2 is delivered to A-Shell by setting DNVALID.
[13]	FATIRQ	FATAL Interrupt Status '0': FATAL interrupt not active '1': FATAL interrupt active, triggered by ARHn_CHCTRL1:FATAL
[12]	CRCERR	CRC Error Status in Upstream Data '0': No CRC error in upstream data (inbound) '1': Indicates occurrence of CRC error in upstream data (inbound)
[11]	CRCTOUT	CRC Timeout Status in Upstream Data '0': No occurrence of CRC timeout in upstream data (inbound) '1': Indicates occurrence of CRC timeout in upstream data (inbound)
[10]	PERROR	Protocol Error Status '0': No occurrence of protocol error '1': Indicates occurrence of protocol error
[9]	READY	A-Shell Ready Status '0': Indicates that A-Shell is not ready to accept outbound trans-actions '1': Indicates that A-Shell is ready to accept outbound transactions
[8]	REMOTERST	Remote A-Shell Restart Status '0': Indicates no restart (transaction realignment procedure) of remote A-Shell is invoked '1': Indicates a restart of remote A-Shell was performed

Table 39-5. ARH Channel Control Register (ARHn_CHCTRL1) Channel 1

Bit Position	Bit Field Name	Bit Description
[7]	UPVALIDCL	Clear Upstream Data Valid Status '0': No effect. Read to this register always reads 0 '1': Clears the upstream data valid status
[6]	DNVALIDST	Set Downstream Data Valid Status '0': No effect. Read to this register always reads 0 '1': Sets the downstream data valid status
[5]	FATIRQCL	Clear Fatal IRQ Interrupt Status '0': No effect. Read to this register always reads 0 '1': Clears the Fatal IRQ interrupt status bit ARHn_CHC-TRL1:FAT- IRQ
[4]	CRCERRCL	Clear CRC Error in Upstream Data Status '0': No effect. Read to this register always reads 0 '1': Clears the CRC error in upstream data (inbound) status bit ARHn_CHCTRL1:CRCERR
[3]	CRCTOUTCL	Clear CRC Timeout in Upstream Data Status '0': No effect. Read to this register always reads 0 '1': Clears the CRC timeout in upstream data (inbound) status bit ARHn_CHCTRL1:CRCOUT
[2]	PERRORCL	Clear Protocol Error Status '0': No effect. Read to this register always reads 0 '1': Clears the protocol error status bit ARHn_CHCTRL1:PERROR
[1]	READYCL	Clear A-Shell Ready Status '0': No effect. Read to this register always reads 0 '1': Clears the A-Shell ready status bit ARHn_CHCTRL1:READY
[0]	REMOTERSTCL	Clear remote A-Shell Restart Status '0': No effect. Read to this register always reads 0 '1': Clears the remote A-Shell restart status bit ARHn_CHCTRL1:REMOTERST

39.2.4 ARH Channel Status Register (ARHn_CHSTAT0 - Channel 0, ARHn_CHSTAT1 - Channel 1)

The software can use this register to read the A-Shell channel status. It contains various status information like upstream CRC error occurrence, upstream synchronization lost, and PLL synchronization lost. Register ARHn_CHSTAT0 for channel 0 is explained here.

ARH Channel Status Register (ARHn_CHSTAT0) Channel 0

Figure 39-4. ARH Channel Status Register (ARHn_CHSTAT0) Channel 0

ARHn_CHSTAT0																															
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**
UPCRC[7]	UPCRC[6]	UPCRC[5]	UPCRC[4]	UPCRC[3]	UPCRC[2]	UPCRC[1]	UPCRC[0]	read0	read0	read0	read0	read0	read0	read0	read0	UPSYNC[7]	UPSYNC[6]	UPSYNC[5]	UPSYNC[4]	UPSYNC[3]	UPSYNC[2]	UPSYNC[1]	UPSYNC[0]	PLLBAD[7]	PLLBAD[6]	PLLBAD[5]	PLLBAD[4]	PLLBAD[3]	PLLBAD[2]	PLLBAD[1]	PLLBAD[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-6. ARH Channel Status Register (ARHn_CHSTAT0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31:24]	UPCRC	CRC Errors Inbound CRC errors. The CRC errors occurred on the data transmission from remote.
[23:16]	read0	-
[15:8]	UPSYNC	Synchronization Lost Indicates number of detected synchronization losses of upstream serial channel (APIX® PHY).
[7:0]	PLLBAD	PLL Synchronization Lost Indicates number of detected PLL synchronization losses.

39.2.5 ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0 - Channel 0, ARHn_CHWDGCTL1 - Channel 1)

This register provides Watchdog Timer control. Using this register generation of Watchdog interrupt can be controlled. Register ARHn_CHWDGCTL0 for channel 0 is explained here.

ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0) Channel 0

Figure 39-5. ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0)

ARHn_CHWDGCTL0																															
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**
WDTXIEN	WDRXIEN	read0	read0	WTTX[1]	WTTX[0]	WTRX[1]	WTRX[0]	read0	read0	read0	read0	read0	read0	read0	read0	WDTXIRQ3	WDTXIRQ2	WDTXIRQ1	WDTXIRQ0	WDRXIRQ3	WDRXIRQ2	WDRXIRQ1	WDRXIRQ0	WDTXIRQ3CL	WDTXIRQ2CL	WDTXIRQ1CL	WDTXIRQ0CL	WDRXIRQ3CL	WDRXIRQ2CL	WDRXIRQ1CL	WDRXIRQ0CL
RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-7. ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0) bits

Bit Position	Bit Field Name	Bit Description
[31]	WDTXIEN	TX Watchdog Timer Interrupt Enable '0': Disables watchdog interrupt for TX '1': Enables Watchdog interrupt for TX
[30]	WDRXIEN	RX Watchdog Timer Interrupt Enable '0': Disables watchdog interrupt for RX, '1': Enables watchdog interrupt for RX
[29:28]	read0	-
[27:26]	WTTX	TX Watchdog Timer Select 00: Select ARNn_CHWDGCTL0:WDTXIRQ0 01: Select ARNn_CHWDGCTL0:WDTXIRQ1 10: Select ARNn_CHWDGCTL0:WDTXIRQ2 11: Select ARNn_CHWDGCTL0:WDTXIRQ3
[25:24]	WTRX	RX Watchdog Timer Select 00: Select ARNn_CHWDGCTL0:WDRXIRQ0 01: Select ARNn_CHWDGCTL0:WDRXIRQ1, 10: Select ARNn_CHWDGCTL0:WDRXIRQ2, 11: Select ARNn_CHWDGCTL0:WDRXIRQ3

Table 39-7. ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0) bits

Bit Position	Bit Field Name	Bit Description
[23:16]	read0	-
[15]	WDTXIRQ3	Watchdog Interrupt '0':No watchdog interrupt for TX generated when watchdog counter value = 2^{19} '1': Watchdog interrupt for TX generated when watchdog counter value = 2^{19}
[14]	WDTXIRQ2	Watchdog Interrupt '0': No watchdog interrupt for TX generated when watchdog counter value = 2^{16} '1': Watchdog interrupt for TX generated when watchdog counter value = 2^{16}
[13]	WDTXIRQ1	Watchdog Interrupt '0': No watchdog interrupt for TX generated when watchdog counter value = 2^{14} '1': Watchdog interrupt for TX generated when watchdog counter value = 2^{14}
[12]	WDTXIRQ0	Watchdog Interrupt '0': No watchdog interrupt for TX generated when watchdog counter value = 2^{13} '1': Watchdog interrupt for TX generated when watchdog counter value = 2^{13}
[11]	WDRXIRQ3	Watchdog Interrupt '0': No watchdog interrupt for RX generated when watch-dog counter value = 2^{19} '1': Watchdog interrupt for RX generated when watchdog counter value = 2^{19}
[10]	WDRXIRQ2	Watchdog Interrupt '0': No watchdog interrupt for RX generated when watch-dog counter value = 2^{18} '1': Watchdog interrupt for RX generated when watchdog counter value = 2^{18}
[9]	WDRXIRQ1	Watchdog Interrupt '0': No watchdog interrupt for RX generated when watch-dog counter value = 2^{17} '1': Watchdog Interrupt for RX generated when watchdog counter value = 2^{17}

Table 39-7. ARH Channel Watchdog Control Register (ARHn_CHWDGCTL0) bits

Bit Position	Bit Field Name	Bit Description
[8]	WDRXIRQ0	Watchdog Interrupt '0': No watchdog interrupt for RX generated when watch-dog counter value = 2^{16} '1': Watchdog interrupt for RX generated when watchdog counter value = 2^{16}
[7]	WDTXIRQ3CL	Clear WDTXIRQ3 '0': No effect. Read to this register always reads 0 '1': Clears WDTXIRQ3
[6]	WDTXIRQ2CL	Clear WDTXIRQ2 '0': No effect. Read to this register always reads 0 '1': Clears WDTXIRQ2
[5]	WDTXIRQ1CL	Clear WDTXIRQ1 '0': No effect. Read to this register always reads 0 '1': Clears WDTXIRQ1
[4]	WDTXIRQ0CL	Clear WDTXIRQ0 '0': No effect. Read to this register always reads 0 '1': Clears WDTXIRQ0
[3]	WDRXIRQ3CL	Clear WDRXIRQ3 '0': No effect. Read to this register always reads 0 '1': Clears WDRXIRQ3
[2]	WDRXIRQ2CL	Clear WDRXIRQ2 '0': No effect. Read to this register always reads 0 '1': Clears WDRXIRQ2
[1]	WDRXIRQ1CL	Clear WDRXIRQ1 '0': No effect. Read to this register always reads 0 '1': Clears WDRXIRQ1
[0]	WDRXIRQ0CL	Clear WDRXIRQ0 '0': No effect. Read to this register always reads 0 '1': Clears WDRXIRQ0

39.2.6 ARH Watchdog Counter Register (ARHn_CHWDGCNT0 - Channel 0, ARHn_CHWDGCNT1 - Channel 1)

The software can use this register to read the status of internal Free Running Timer. Register ARHn_CHWDGCNT0 for channel 0 is explained here.

ARH Watchdog Counter Register (ARHn_CHWDGCNT0) Channel 0

Figure 39-6. ARH Watchdog Counter Register (ARHn_CHWDGCNT0) Channel 0

ARHn_CHWDGCNT0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CNT[15]	CNT[14]	CNT[13]	CNT[12]	CNT[11]	CNT[10]	CNT[9]	CNT[8]	CNT[7]	CNT[6]	CNT[5]	CNT[4]	CNT[3]	CNT[2]	CNT[1]	CNT[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 39-8. ARH Watchdog Counter Register (ARHn_CHWDGCNT0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	CNT	Free Running Timer Reading this register provides the value of upper 16 bits of the 20 bits Free Running Timer.

39.2.7 ARH Transaction Buffer Control Register (ARHn_TBCTRL0~15)

The software can use these registers (ARHn_TBCTRL0 to ARHn_TBCTRL15) to program the Transaction Buffer. This register can be written when the corresponding Transaction Buffer state is ARHn_TBCTRL0:ACTIVE = '0', ARHn_TBCTRL0:WAITING = '0', and ARHn_TBCTRL0:PENDING = '0'. Register for ARHn_TBCTRL0 is explained here.

ARH Transaction Buffer Control Register (ARHn_TBCTRL0)

Figure 39-7. ARH Transaction Buffer Control Register (ARHn_TBCTRL0)

ARHn_TBCTRL0																															
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**
TBCH	TBAINC	TBACT	TBIMD	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TBDEN	TBIEN	read0	read0	ACTIVE	UNLOCKED	CANCELED	WAITING	PENDING	TBIRQ	read0	read0	read0	read0	read0	UNLOCKEDCL	CANCELEDCL	TBIRQCL
RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-9. ARH Transaction Buffer Control Register (ARHn_TBCTRL0) bits

Bit Position	Bit Field Name	Bit Description
[31]	TBCH	Transaction Buffer Channel '0': Transaction Buffer assigned to channel 0 '1': Transaction Buffer assigned to channel 1
[30]	TBAINC	Transaction Buffer Address Increment '0': Transaction Buffer Address Increment is disabled, '1': Transaction Buffer Address Increment is enabled **Notes:** Increments address after WR access of TFDATA and transmission of TF. Increments address after reception of requested TF and RD access to TFDATA.
[29]	TBACT	Transaction Buffer Activated '0': Transaction Buffer is activated by WR access to ARHn_RHCTRL:TBNO, '1': Transaction Buffer is activated by WR access to ARHn_RHCTRL:TBNO or TFDATA (RD or WR)

Table 39-9. ARH Transaction Buffer Control Register (ARHn_TBCTRL0) bits

Bit Position	Bit Field Name	Bit Description
[28]	TBIMD	Transaction Buffer Interrupt Mode '0': Transaction Buffer interrupt on TB idle (after transaction send) '1': Transaction Buffer interrupt on TB valid (after read request data reception)
[27:18]	read0	-
[17]	TBDEN	Transaction Buffer DMA Enable '0': Transaction Buffer DMA is disabled, '1': Transaction Buffer DMA is enabled
[16]	TBIEN	Transaction Buffer Interrupt Enable '0': Transaction Buffer interrupt is disabled, '1': Transaction Buffer interrupt is enabled
[15:14]	read0	-
[13]	ACTIVE	Transaction Buffer Request Active '0': No active data in Transaction Buffer, '1': Active data in Transaction Buffer (delivery to A-Shell requested)
[12]	UNLOCKED	Transaction Buffer Unlocked '0': No last action on this buffer, '1': Last action on this Transaction Buffer was unlock
[11]	CANCELED	Transaction Buffer Request Cancelled '0': No last action on this buffer, '1': Last action on this Transaction Buffer was a successful cancel
[10]	WAITING	Transaction Buffer Request Waiting '0': Not waiting for requested data '1': Transaction Buffer waiting for requested data **Note:** Waiting is cleared at reception of requested data in buffer.
[9]	PENDING	Transaction Buffer Request Pending '0': No pending data in Transaction Buffer '1': Pending data in Transaction Buffer (not yet requested delivery to A-Shell)
[8]	TBIRQ	Transaction Buffer Interrupt '0': Transaction Buffer interrupt not active, '1': Transaction Buffer interrupt active TBIRQ interrupt generated due to sending WR msg or reception of RD msg.
[7:3]	read0	-
[2]	UNLOCKEDCL	Clear Transaction Buffer Unlocked '0': No effect. Read to this register always reads '0' '1': Clears the Transaction Buffer unlocked status

Table 39-9. ARH Transaction Buffer Control Register (ARHn_TBCTRL0) bits

Bit Position	Bit Field Name	Bit Description
[1]	CANCELEDCL	Clear Transaction Buffer Cancelled '0': No effect. Read to this register always reads '0' '1': Clears the Transaction Buffer cancelled status
[0]	TBIRQCL	Clear Transaction Buffer Interrupt '0': No effect. Read to this register always reads '0' '1': Clears Transaction Buffer interrupt ARHn_TBCTRL0:TBIRQ

Figure 39-8. Example showing Transaction Buffer states for frame transmission

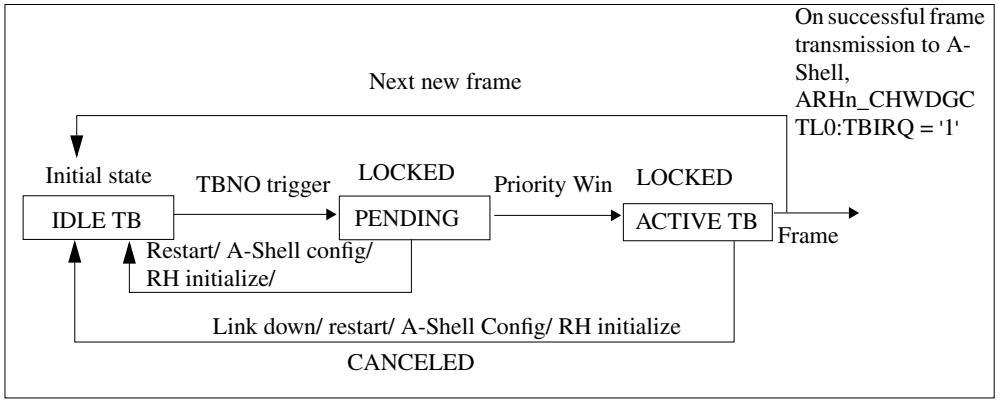
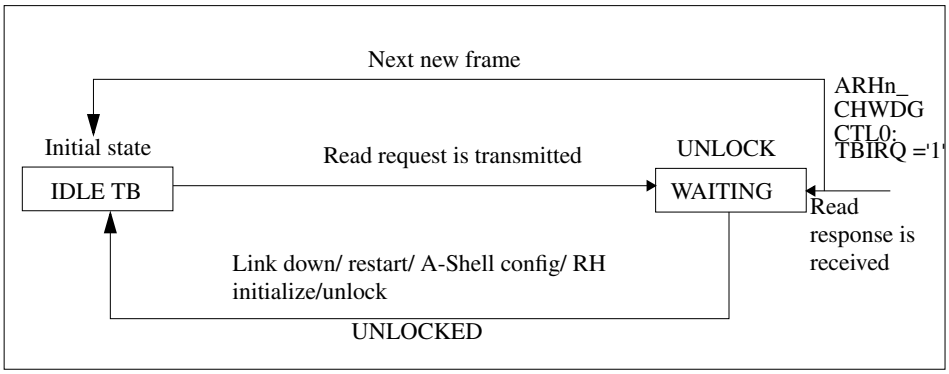


Figure 39-9. Example showing Transaction Buffer states for frame reception



Note:

- Only ARHn_TBCTRL0:WAITING Transaction Buffers can be UNLOCKED
- Only ARHn_TBCTRL0:PENDING or ARHn_TBCTRL0:ACTIVE Transaction Buffers can be CANCELLED
- While in ARHn_TBCTRL0:PENDING or ARHn_TBCTRL0:ACTIVE states, the Transaction Buffer is LOCKED Figure 39-8 explains Transaction Buffer states during frame transmission.

1. Writing ARHn_RHCTRL:TBNO Transaction Buffer when ARHn_RHCTRL:UNLOCK = '0' and ARHn_RHCTRL:CANCEL = '0' moves the Transaction Buffer to pending/active state based on priority and if no transmission is pending.
2. A Transaction Buffer in pending/active state will revert back to idle state upon receipt of a ARHn_RHCTRL:CANCEL request or RESTART request.
3. A Transaction Buffer in active state will revert back to idle state after transmission of frame.

Figure 39-9 explains Transaction Buffer states during frame reception.

1. A Transaction Buffer in idle state goes to waiting state on read request submission to remote device.
2. A Transaction Buffer in waiting state will revert back to idle upon ARHn_RHCTRL:UNLOCK, RESTART, or ARHn_CHC-TRL0:INITRH request.
3. A Transaction Buffer in waiting state will revert back to idle state on reception of read response from remote device.

39.2.8 ARH Transaction Buffer Interrupt Register (ARHn_TBIRQ)

The software can read this register to get the status of Transaction Buffer interrupt of all the Transaction Buffers.

ARH Transaction Buffer Interrupt Register (ARHn_TBIRQ)

Figure 39-10. ARH Transaction Buffer Interrupt Register (ARHn_TBIRQ)

ARHn_TBIRQ															
15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TBIRQ[15]	TBIRQ[14]	TBIRQ[13]	TBIRQ[12]	TBIRQ[11]	TBIRQ[10]	TBIRQ[9]	TBIRQ[8]	TBIRQ[7]	TBIRQ[6]	TBIRQ[5]	TBIRQ[4]	TBIRQ[3]	TBIRQ[2]	TBIRQ[1]	TBIRQ[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-10. ARH Transaction Buffer Interrupt Register (ARHn_TBIRQ) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	TBIRQ	Transaction Buffer Interrupt Read-only TB interrupt flag (ARHn_TBIRQ:TBIRQ[n]) is generated when interrupt enable for 15-Transcation Buffer[n] is high (ARHn_TBCTRL15~00:TBIEN == '1'). **Notes:** TBIRQ can or is cleared by the following events: 1. Cleared by SW on write access to ARHn_TBCTRL15~00:TBIRQCL flag with data '1'. 2. Cleared by HW if ARHn_TBCTRL15~00:TBACT == '1' and read or write access to ARHn_TFDATA register (both CPU or DMA). **Note:** ARHn_TBIRQ[15] corresponds to Transaction buffer[0]'s interrupt and ARHn_TBIRQ[0] corresponds to Transaction buffer[15]'s interrupt.

39.2.9 ARH Transaction Buffer Index Register (ARHn_TBIDX0 - Channel 0, ARHn_TBIDX1 - Channel 1)

The software can use this register to find the index of currently active Transaction Buffer with active ARHn_TBCTRL0:TBIRQ and ARHn_TBCTRL0:TBIEN interrupt. The ARHn_TBIDX0 register is for channel 0, while ARHn_TBIDX1 register is for channel 1. Register ARHn_TBIDX0 for channel 0 is explained here.

ARH Transaction Buffer Index Register (ARHn_TBIDX0) Channel 0

Figure 39-11. ARH Transaction Buffer Index Register (ARHn_TBIDX0) Channel 0

ARHn_TBIDX0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
read0	read0	read0	TBIDX[4]	TBIDX[3]	TBIDX[2]	TBIDX[1]	TBIDX[0]
Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0

Table 39-11. ARH Transaction Buffer Index Register (ARHn_TBIDX0) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[7:5]	read0	-
[4:0]	TBIDX	Transaction Buffer Index These bits provide read-only index + 1 of highest priority buffer with active (ARHn_TBCTRL00~15:TBIRQ == '1') and enabled (ARHn_TBCTRL00~15:TBIEN == '1') interrupt. If TBIDX == '0' indicates no interrupt for corresponding channel.

39.2.10 ARH Transaction Frame Control Register (ARHn_TFCTRL00~15)

The software can use these registers (ARHn_TFCTRL0 to ARHn_TFCTRL15) to program the Transaction Frame. These registers can be written when the corresponding Transaction Buffer state is ARHn_TBCTRL0:ACTIVE = '0', ARHn_TBCTRL0:WAITING = '0', and ARHn_TBCTRL0:PENDING = '0'. Register ARHn_TFCTRL0 is explained here.

ARH Transaction Frame Control Register (ARHn_TFCTRL0)

Figure 39-12. ARH Transaction Frame Control Register (ARHn_TFCTRL0)

ARHn_TFCTRL0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
TFDASWP	TFAINV	read0	ERROR	SZ[1]	SZ[0]	OAEN	RW
RpWp	RpWp	Rp0	Rp	RWP	RWP	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 39-12. ARH Transaction Frame Control Register (ARHn_TFCTRL0) bits

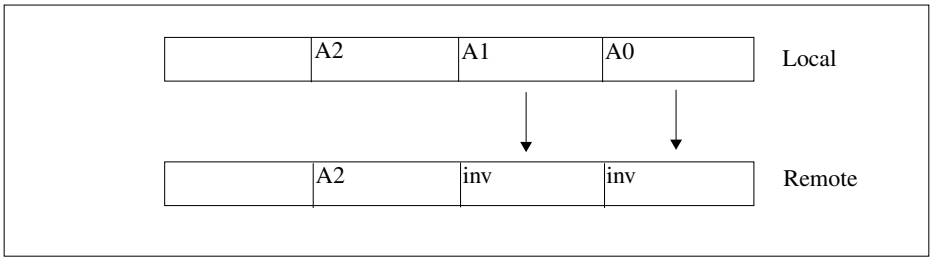
Bit Position	Bit Field Name	Bit Description
[7]	TFDASWP	Transaction Frame Data Swap '0': Normal mode '1': Byte swapping For more details refer to Figure 39-14.
[6]	TFAINV	Transaction Frame Address Inversion '0': Normal mode '1': Address inversion For more details refer to Figure 39-15.
[5]	read0	-
[4]	ERROR	Remote Handler RX Error '0': Normal operation '1': Remote Handler RX bus error occurred
[3:2]	SZ	Size '00': Byte, '01': Halfword, '10': Word, '11': Reserved
[1]	OAEN	Offset Address Enable '0': Offset address is disabled '1': Offset address is enabled

Table 39-12. ARH Transaction Frame Control Register (ARHn_TFCTRL0) bits

Bit Position	Bit Field Name	Bit Description
[0]	RW	Read or Write '0': Read message to be sent '1': Write message to be sent

Note: In address inversion mode the two least significant bits of the address are inverted.

Figure 39-13. Example showing address inversion (ARHn_TFCTRL00~15:TFAINV = '1')



In swapping mode depending on the configured size the following byte swapping of the data is as shown in Figure 39-15.

Figure 39-14. ARH TFDATA byte swapping depending on swapping mode and configured size

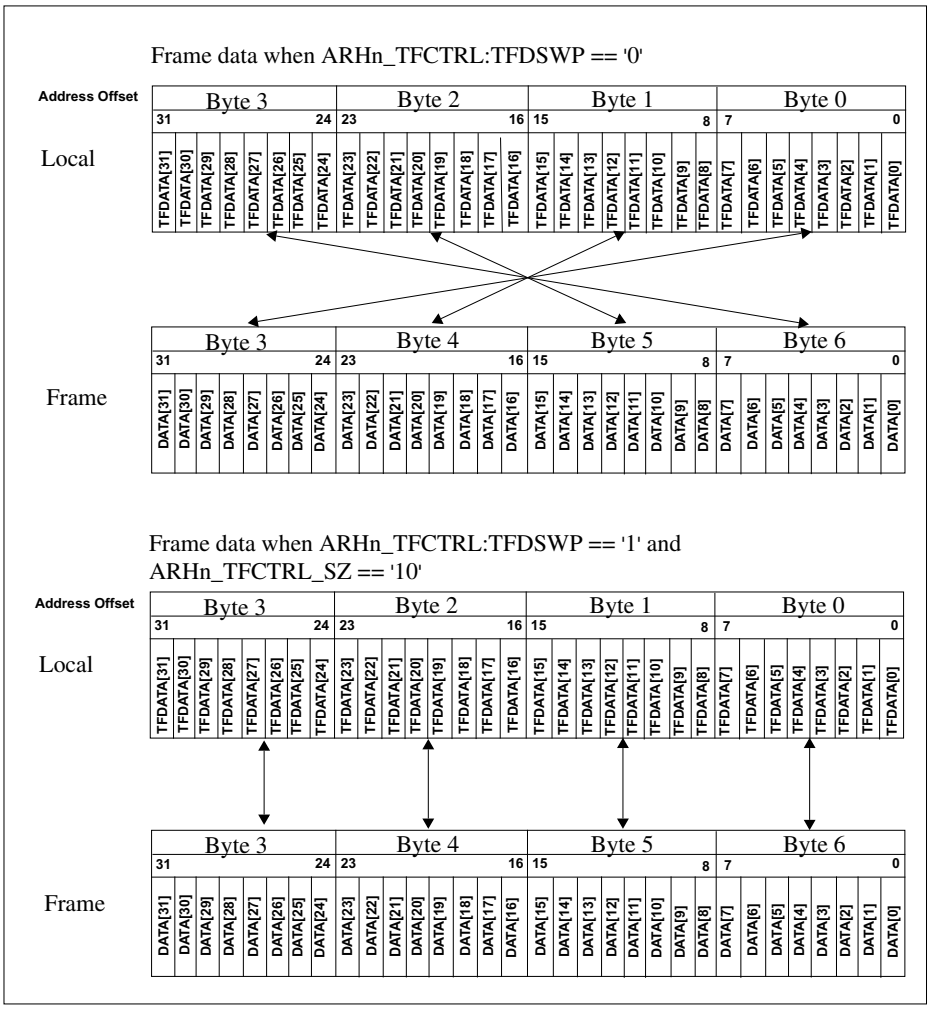
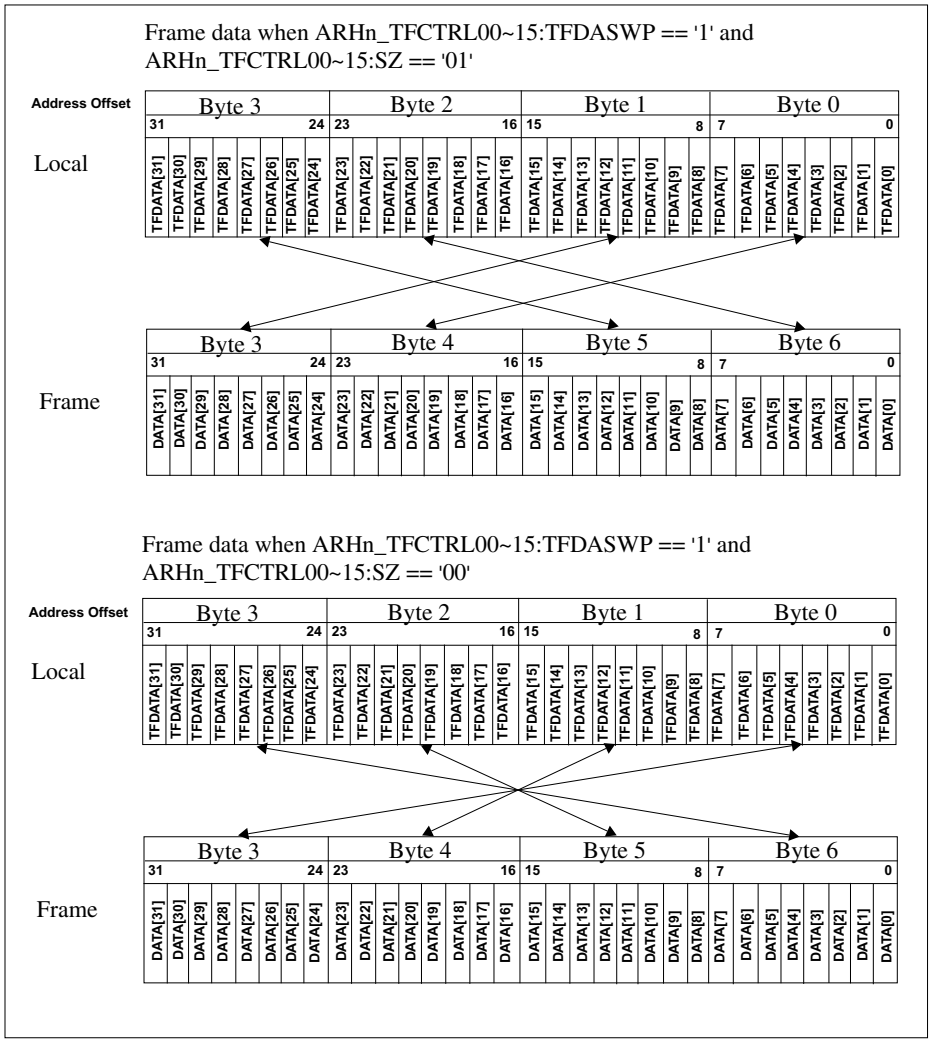


Figure 39-15. ARH TFDATA byte swapping depending on swapping mode and configured size



39.2.11 ARH Transaction Frame Index Register (ARHn_TFIDX0~15)

The software can use these registers (ARHn_TFIDX0 to ARHn_TFIDX15) to program the index of the Transaction Frame. These registers can be written when the corresponding Transaction Buffer state is ARHn_TBCTRL0:ACTIVE = '0', ARHn_TBCTRL0:WAITING = '0', and ARHn_TBCTRL0:PENDING = '0'. Register ARHn_TFIDX0 is explained here.

ARH Transaction Frame Index Register (ARHn_TFIDX0)

Figure 39-16. ARH Transaction Frame Index Register (ARHn_TFIDX0)

ARHn_TFIDX0							
07	**06**	**05**	**04**	**03**	**02**	**01**	**00**
IDX[7]	IDX[6]	IDX[5]	IDX[4]	IDX[3]	IDX[2]	IDX[1]	IDX[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0

Table 39-13. ARH Transaction Frame Index Register (ARHn_TFIDX0) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	IDX	Transaction Frame Index Any number between 0 and 255. **Note:** Index is used for read request. Received data from a read request is stored in an active Transaction Buffer with matching index. If there is no active Transaction Buffer with matching index (e.g. by using unlock), the received data is discarded.

39.2.12 ARH Transaction Frame Address Register (ARHn_TFADDR0~15)

The software can use these registers (ARHn_TFADDR0 to ARHn_TFADDR15) to program the address of Transaction Frame. This register can be written when the corresponding Transaction Buffer state is ARHn_TBCTRL0:ACTIVE = '0', ARHn_TBCTRL0:WAITING = '0', and ARHn_TBCTRL0:PENDING = '0'. Register ARHn_TFADDR0 is explained here.

ARH Transaction Frame Address Register (ARHn_TFADDR0)

Figure 39-17. ARH Transaction Frame Address Register (ARHn_TFADDR0)

ARHn_TFADDR0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ADDR[19]	ADDR[18]	ADDR[17]	ADDR[16]	ADDR[15]	ADDR[14]	ADDR[13]	ADDR[12]	ADDR[11]	ADDR[10]	ADDR[9]	ADDR[8]	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-14. ARH Transaction Frame Address Register (ARHn_TFADDR0) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:20]	read0	-
[19:0]	ADDR	Transaction Frame Address Transaction Frame Address.

39.2.13 ARH Transaction Frame Data Register (ARHn_TFDATA0~15)

The software can use these registers (ARHn_TFDATA0 to ARHn_TFDATA15) to program the Transaction Frame data of Transaction Frame. This register can be written when the corresponding Transaction Buffer state is ARHn_TBCTRL0:ACTIVE = '0', ARHn_TBCTRL0:WAITING = '0', and ARHn_TBCTRL0:PENDING = '0'. Register ARHn_TFDATA0 is explained here.

ARH Transaction Frame Data Register (ARHn_TFDATA0)

Figure 39-18. ARH Transaction Frame Data Register (ARHn_TFDATA0)

ARHn_TFDATA0																															
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**
TFDATA[31]	TFDATA[30]	TFDATA[29]	TFDATA[28]	TFDATA[27]	TFDATA[26]	TFDATA[25]	TFDATA[24]	TFDATA[23]	TFDATA[22]	TFDATA[21]	TFDATA[20]	TFDATA[19]	TFDATA[18]	TFDATA[17]	TFDATA[16]	TFDATA[15]	TFDATA[14]	TFDATA[13]	TFDATA[12]	TFDATA[11]	TFDATA[10]	TFDATA[9]	TFDATA[8]	TFDATA[7]	TFDATA[6]	TFDATA[5]	TFDATA[4]	TFDATA[3]	TFDATA[2]	TFDATA[1]	TFDATA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-15. ARH Transaction Frame Data Register (ARHn_TFDATA0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TFDATA	Transaction Frame Data Transaction Frame Data.

Notes:

ARHn_TFDATA0~15 Registers Read Conditions:

1. For a DAP Controller master, read is allowed if there is no PPU violation.
2. For a AHB master other than DAP Controller,
   a. If ARHn_TBCTRLn:TBACT = ‘1’ and if the transaction buffer state is ARHn_TBCTRLn:ACTIVE = ‘1’ or ARHn_TBCTRLn:WAITING = ‘1’ or ARHn_TBCTRLn:PENDING = ‘1’, error response is generated in both privileged as well as non-privileged mode.
   b. In non-privileged mode, if ARHn_TBCTRLn:TBACT = ‘1’ and if the transaction buffer state is ARHn_TBCTRLn:ACTIVE = ‘0’ and ARHn_TBCTRLn:WAITING = ‘0’ and ARHn_TBCTRLn:PENDING = ‘0’, read is allowed only if PPU access = ‘1’.
   c. For all other conditions, read is allowed if there is no PPU violation.

ARHn_TFDATA0~15 Registers Write Conditions:

1. Write is not allowed if the transaction buffer state is ARHn_TBCTRLn:ACTIVE = ‘1’ or ARHn_TBCTRLn:WAITING = ‘1’ or ARHn_TBCTRLn:PENDING = ‘1’ in both privileged as well as non-privileged mode.
2. For all other conditions in non-privileged mode, write is allowed if there is no PPU violation.

Transaction Buffer Trigger Conditions:

1. Read transaction from a DAP Controller master does not trigger transaction buffer.
2. For a read transaction from a AHB master other than DAP Controller or a write transaction from any AHB master,

a. In privileged mode, transaction buffer is triggered only if ARHn_TBCTRLn:TBACT = ‘1’ and the transaction buffer state is ARHn_TBCTRLn:ACTIVE = ‘0’ and ARHn_TBCTRLn:WAITING = ‘0’ and ARHn_TBCTRLn:PENDING = ‘0’.
b. In non-privileged mode, transaction buffer is triggered only if ARHn_TBCTRLn:TBACT = ‘1’ and the transaction buffer state is ARHn_TBCTRLn:ACTIVE = ‘0’ and ARHn_TBCTRLn:WAITING = ‘0’ and ARHn_TBCTRLn:PENDING = ‘0’ and PPU access = ‘1’.

39.2.14 ARH Event Control Register (ARHn_EVCTRL)

The software can use this register to program the Event Buffer. This register can be written when ARHn_RHCTRL:LST = '0'.

ARH Event Control Register (ARHn_EVCTRL)

Figure 39-19. ARH Event Control Register (ARHn_EVCTRL)

ARHn_EVCTRL																													
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**
MODE	FRST	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	STATUS[7]	STATUS[6]	STATUS[5]	STATUS[4]	STATUS[3]	STATUS[2]	STATUS[1]	STATUS[0]	LEVEL[7]	LEVEL[6]	LEVEL[5]	LEVEL[4]	LEVEL[3]	LEVEL[2]	LEVEL[1]	LEVEL[0]
RpWp	Rp0Wp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 39-16. ARH Event Control Register (ARHn_EVCTRL) bits

Bit Position	Bit Field Name	Bit Description
[31]	MODE	FIFO Mode '0': Level mode, on FIFO full condition new events are discarded '1': Ring mode, oldest event is overwritten with latest event (circular buffering)
[30]	FRST	FIFO Reset '0': FIFO is in normal operation, '1': FIFO pointers are reset pulse (set to '0' after 1 cycle)
[29:24]	read0	-
[23:16]	read0	-
[15:8]	STATUS	FIFO Fill Status Current FIFO filling status read-only, the FIFO fill status ranges from 0 to 128.
[7:0]	LEVEL	FIFO Interrupt Level FIFO interrupt level (128 default). The software can program any value in range of 0 to 128.

39.2.15 ARH Event Interrupt Control Register (ARHn_EVIRQC)

The software can use this register to program the event interrupts. This register provides enable and status information for the level, overflow, and Event Buffer interrupts.

ARH Event Interrupt Control Register (ARHn_EVIRQC)

Figure 39-20. ARH Event Interrupt Control Register (ARHn_EVIRQC)

ARHn_EVIRQC																															
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**
LVIEN	OFLIEN	EVIEN	read0	read0	read0	read0	read0	LVIRQ	OFLIRQ	EVIRQ	read0	read0	read0	read0	read0	LVIRQCL	OFLIRQCL	EVIRQCL	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0Wp	Rp0Wp	Rp0Wp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-17. ARH Event Interrupt Control Register (ARHn_EVIRQC) bits

Bit Position	Bit Field Name	Bit Description
[31]	LVIEN	Level Interrupt Enable '0': Level interrupt is disabled '1': Level interrupt is enabled
[30]	OFLIEN	Event Buffer Overflow Interrupt Enable '0': Event Buffer overflow interrupt not active '1': Event Buffer overflow interrupt active
[29]	EVIEN	Event Buffer Interrupt Enable '0': Event Buffer interrupt is disabled '1': Event Buffer interrupt is enabled
[28:24]	read0	-
[23]	LVIRQ	Level IRQ '0': Level interrupt not active '1': Level interrupt active (if STATUS >= LEVEL)
[22]	OFLIRQ	Overflow IRQ '0': Event Buffer overflow interrupt not active '1': Event Buffer interrupt active
[21]	EVIRQ	Event IRQ '0': Event Buffer interrupt is disabled '1': Event Buffer interrupt is enabled
[20:16]	read0	-

Table 39-17. ARH Event Interrupt Control Register (ARHn_EVIRQC) bits

Bit Position	Bit Field Name	Bit Description
[15]	LVIRQCL	Clear Level IRQ '0': No effect. Read to this register always reads '0' '1': Clears LVIRQ
[14]	OFLIRQCL	Clear Event Buffer Overflow IRQ '0': No effect. Read to this register always reads '0' '1': Clears OFLIRQ
[13]	EVIRQCL	Clear Event Buffer Event IRQ '0': No effect. Read to this register always reads '0' '1': Clears EVIRQ
[12:8]	read0	-
[7:0]	read0	-

39.2.16 ARH Event Buffer Register (ARHn_EVBUF0)

The software can use this register to program the index number of event messages. It also provides status of channel number on which event was last received. This register can be written when ARHn_TST:TM = '1'.

ARH Event Buffer Register (ARHn_EVBUF0)

Figure 39-21. ARH Event Buffer Register (ARHn_EVBUF0)

ARHn_EVBUF0																															
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**
read0	read0	read0	read0	read0	read0	read0	EVCH	EVIDX[7]	EVIDX[6]	EVIDX[5]	EVIDX[4]	EVIDX[3]	EVIDX[2]	EVIDX[1]	EVIDX[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-18. ARH Event Buffer Register (ARHn_EVBUF0) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	EVCH	Event Channel Number Holds channel number from Remote Handler RX (remote device) event.
[23:16]	EVIDX	Event Index Number 0-255 holds index number from Remote Handler RX event.
[15:0]	read0	-

Notes:

1. A read access to ARHn_EVBUF0 triggers a retrieve of the current event message from the Event Buffer FIFO and returns the channel number and event index. Therefore, if ARHn_TST:TM == '0' non-DAP access in non-privileged mode can only be done with full access rights (i.e. when iPPU_ACCESS == '1'), else protection violation fault is triggered. However, if the DAP controller reads this register, the Event Buffer FIFO read pointer is not incremented.
2. It is recommended to read first ARHn_EVBUF0 and after that ARHn_EVBUF1 registers.

39.2.17 ARH Event Buffer Register (ARHn_EVBUF1)

The software can use this register to read the 4 byte payload data of event message. This register can be written when ARHn_TST:TM = '1'.

ARH Event Buffer Register (ARHn_EVBUF1)

Figure 39-22. ARH Event Buffer Register (ARHn_EVBUF1)

ARHn_EVBUF1																															
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**
EVDATA[31]	EVDATA[30]	EVDATA[29]	EVDATA[28]	EVDATA[27]	EVDATA[26]	EVDATA[25]	EVDATA[24]	EVDATA[23]	EVDATA[22]	EVDATA[21]	EVDATA[20]	EVDATA[19]	EVDATA[18]	EVDATA[17]	EVDATA[16]	EVDATA[15]	EVDATA[14]	EVDATA[13]	EVDATA[12]	EVDATA[11]	EVDATA[10]	EVDATA[9]	EVDATA[8]	EVDATA[7]	EVDATA[6]	EVDATA[5]	EVDATA[4]	EVDATA[3]	EVDATA[2]	EVDATA[1]	EVDATA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 39-19. ARH Event Buffer Register (ARHn_EVBUF1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EVDATA	Payload Data The 4 byte payload data. A read access to ARHn_EVBUF1 returns the data part of the event message.

Note: It is recommended to read first ARHn_EVBUF0 register and after that ARHn_EVBUF1 register.

39.2.18 ARH APIX® Configuration Register (ARHn_APCFG00 - Channel 0)

The software can use this register to program the APIX® PHY. This register ARHn_APCFG00 can be written when ARHn_CHCTRL0:TXCFG = '1'.

ARH APIX® Configuration Register (ARHn_APCFG00) Channel 0

Figure 39-23. ARH APIX® Configuration Register (ARHn_APCFG00) Channel 0

ARHn_APCFG00																															
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**
PWRDOWN	PWRDOWNPREEMP	PWRDOWNNOM	PWRDOWNUP	reserved	DWNPREEMPCTRL[2]	DWNPREEMPCTRL[1]	DWNPREEMPCTRL[0]	DWNBWMODE[1]	DWNBWMODE[0]	PXINCLKACTIVEEDGE	PXINJUMBLEEN	PXINCTRLPIGGYBACK[1]	PXINCTRLPIGGYBACK[0]	PXDATAWIDTH[1]	PXDATAWIDTH[0]	reserved	UPBWMODE[1]	UPBWMODE[0]	UPLENIENTWINDOW	UPSMPOFST[3]	UPSMPOFST[2]	UPSMPOFST[1]	UPSMPOFST[0]	JUMBLERMASKEN	SBUPSKIPSKIP	SBUPRESTARTONERROR	BOOTSUPPORTDISABLE	TRACEFEEDENABLE	CEB6DISABLE	CEB4DISABLE	IGNOREPLLGOOD
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

Table 39-20. ARH APIX® Configuration Register (ARHn_APCFG00) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31]	PWRDOWN	APIX® PHY Global power down (upstream, downstream and PLL) '0': Power Down '1': Power Up
[30]	PWRDOWNPREEMP	APIX® PHY Power down pre-emphasis serializer and pre-emphasis output driver (diff amp) '0': Power Down '1': Power Up
[29]	PWRDOWNNOM	APIX® PHY Power down nominal serializer and nominal output driver (diff amp) '0': Power Down '1': Power Up
[28]	PWRDOWNUP	APIX® PHY Power down upstream path '0': Power Down '1': Power Up
[27]	reserved	Reserved

Table 39-20. ARH APIX® Configuration Register (ARHn_APCFG00) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[26:24]	DWNPREEMPCTRL	APIX® PHY Pre-emphasis Configuration Reduce output current (pre-emphasis) after N equal serial bits (N = 0..7)
[23:22]	DWNBWMODE	APIX® PHY TX Mode Select '00': 125 Mbit/s (Low Bandwidth Mode 1), '01': 250 Mbit/s (Low Bandwidth Mode 2) '10': 500 Mbit/s (Half Bandwidth Mode) '11': 1000 Mbit/s (Full Bandwidth Mode)
[21]	PXINCLKACTIVEEDGE	A-Shell Active Clock Edge Configuration Clock Edge at which pixel data and pixel controls are captured. '0': Falling edge, '1': Rising edge
[20]	PXINJUMBLEEN	A-Shell Cyclic Pixel Data Inversion Configuration '0': Disable, '1': Enable
[19:18]	PXINCTRLPIGGYBACK	A-Shell Pixel Control Data Transmission Configuration '00': Never, '01': Unused '10': With even pixels only '11': With every pixel,
[17:16]	PXDATAWIDTH	A-Shell Pixel Data Width Configuration '00': 10 bits, '01': 12 bits '10': 18 bits, '11': 24 bits
[15]	reserved	Reserved
[14:13]	UPBWMODE	APIX® PHY Master Upstream Data Rate '00': 62.50 MBit/s (Full, Half, Low Bandwidth Mode) '01': 41.67 MBit/s (Full Bandwidth Mode) '10': 31.25 MBit/s (Full, Half, Low Bandwidth Mode) '11': 20.83 MBit/s (Half and Low Bandwidth Mode)
[12]	UPLENIENTWINDOW	A-Shell Synchronization Error Tolerance Configuration Tolerates a single synchronization error within a defined timing window without automatic re-synchronization. '0': Disable, '1': Enable

Table 39-20. ARH APIX® Configuration Register (ARHn_APCFG00) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[11:8]	UPSMPOFST	APIX® PHY Upstream Sampling Point Configuration '0000': Optimum sampling point when operating in 62.50 Mbit/s mode, '0010': Optimum sampling point when operating in 41.67 MBit/s or 31.25 MBit/s mode '0100': Optimum sampling point when operating in 20.83 MBit/s mode
[7]	JUMBLERMASKEN	A-Shell Pixel Data Mask Enable Configuration Masks Pixel Data. '0': Disable, '1': Enable
[6]	SBUPSKIPSKIP	A-Shell Resynchronization in Upstream Channel Configuration This bit accelerates resynchronization after loss of synchronization by assuming that the upstream receiver is still locked (but bits are corrupted). '0': Enable, '1': Disable
[5]	SBUPRESTARTONERROR	A-Shell Restart on Error in Upstream Channel Configuration This bit forces re-alignment at bit level in upstream channel whenever synchronization is lost. '0': Disable, '1': Enable
[4]	BOOTSUPPORTDISABLE	A-Shell Boot Support Disable Configuration This bit specifies the condition when pixel data synchronization with the receiver is requested. '0': Wait for the rising edge of pixel data enable, '1': Wait for pixel data enable == '1'
[3]	TRACEFEEDENABLE	A-Shell Trace Feed Enable Configuration This bit enables the detection of discontinued pixel stream (pixel clock) or frequency below 4 MHz. '0': Disable, '1': Enable
[2]	CEB6DISABLE	A-Shell Consecutive 6 Bits Equivalence Check Configuration Check for 6 consecutive bits with equal value. '0': Enable, '1': Disable,

Table 39-20. ARH APIX® Configuration Register (ARHn_APCFG00) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[1]	CEB4DISABLE	A-Shell Consecutive 4 Bits Equivalence Check Configuration Check for 4 consecutive bits with equal value. '0': Enable, '1': Disable,
[0]	IGNOREPLLGOOD	APIX® PHY PLLGOOD Output Ignore Configuration Loss of PLL's synchronization to the reference results in a hard-ware reset. '0': Enable, '1': Disable,

39.2.19 ARH APIX® Configuration Register (ARHn_APCFG10 - Channel 1)

The software can use this register to program the APIX® PHY. This register can be written when ARHn_CHCTRL1:TXCFG = '1'.

ARH APIX® Configuration Register (ARHn_APCFG10) Channel 1

Figure 39-24. ARH APIX® Configuration Register (ARHn_APCFG10) Channel 1

ARHn_APCFG10																															
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
unused[7]	unused[6]	unused[5]	unused[4]	unused[3]	unused[2]	unused[1]	unused[0]	reserved[1]	reserved[0]	PXINCLKACTIVEEDGE	PXINJUMBLEEN	PXINCTRLPIGGYBACK[1]	PXINCTRLPIGGYBACK[0]	PXDATAWIDTH[1]	PXDATAWIDTH[0]	reserved[2]	reserved[1]	reserved[0]	UPLENIENTWINDOW	reserved[3]	reserved[2]	reserved[1]	reserved[0]	JUMBLERMASKEN	SBUPSKIPSKIP	SBUPRESTARTONERROR	BOOTSUPPORTDISABLE	TRACEFEEDENABLE	CEB6DISABLE	CEB4DISABLE	IGNOREPLLGOOD
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

Table 39-21. ARH APIX® Configuration Register (ARHn_APCFG10) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[31:24]	unused	Unused
[23:22]	reserved	Reserved
[21]	PXINCLKACTIVEEDGE	A-Shell Active Clock Edge Configuration Clock Edge at which pixel data and pixel controls are captured. '0': Falling edge, '1': Rising edge
[20]	PXINJUMBLEEN	A-Shell Cyclic Pixel Data Inversion Configuration '0': Disable, '1': Enable
[19:18]	PXINCTRLPIGGYBACK	A-Shell Pixel Control Data Transmission Configuration '00': Never, '01': Unused '10': With even pixels only '11': With every pixel,

Table 39-21. ARH APIX® Configuration Register (ARHn_APCFG10) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[17:16]	PXDATAWIDTH	A-Shell Pixel Data Width Configuration '00': 10 bits, '01': 12 bits '10': 18 bits, '11': 24 bits
[15:13]	reserved	Reserved
[12]	UPLENIENTWINDOW	A-Shell Synchronization Error Tolerance Configuration Tolerates a single synchronization error within a defined timing window without automatic re-synchronization. '0': Disable, '1': Enable
[11:8]	reserved	Reserved
[7]	JUMBLERMASKEN	A-Shell Pixel Data Mask Enable Configuration Masks Pixel Data. '0': Disable, '1': Enable
[6]	SBUPSKIPSKIP	A-Shell Resynchronization in Upstream Channel Configuration This bit accelerates resynchronization after loss of synchronization by assuming that the upstream receiver is still locked (but bits are corrupted). '0': Enable, '1': Disable
[5]	SBUPRESTARTONERROR	A-Shell Restart on Error in Upstream Channel Configuration This bit forces realignment at bit Level in upstream channel whenever synchronization is lost. '0': Disable '1': Enable
[4]	BOOTSUPPORTDISABLE	A-Shell Boot Support Disable Configuration This bit specifies the condition when pixel data synchronization with the receiver is requested. '0': Wait for rising edge of pixel data enable, '1': Wait for pixel data enable == '1'
[3]	TRACEFEEDENABLE	A-Shell Trace Feed Enable Configuration This bit enables the detection of discontinued pixel stream (pixel clock) or frequency below 4 MHz. '0': Enable, '1': Disable

Table 39-21. ARH APIX® Configuration Register (ARHn_APCFG10) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[2]	CEB6DISABLE	A-Shell Consecutive 6 Bits Equivalence Check Configuration Check for 6 consecutive bits with equal value. '0': Enable '1': Disable
[1]	CEB4DISABLE	A-Shell Consecutive 4 Bits Equivalence Check Configuration Check for 4 consecutive bits with equal value. '0': Enable '1': Disable
[0]	IGNOREPLLGOOD	APIX® PHY PLLGOOD Output Ignore Configuration Loss of PLL's synchronization to the reference results in a hardware reset. '0': Enable '1': Disable

39.2.20 ARH APIXR Configuration Register (ARHn_APCFG01 - Channel 0)

The software can use this register to program the APIX® PHY. This register can be written when ARHn_CHCTRL0:TXCFG = '1'.

ARH APIX® Configuration Register (ARHn_APCFG01) Channel 0

Figure 39-25. ARH APIX® Configuration Register (ARHn_APCFG01) Channel 0

ARHn_APCFG01																															
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**
DDOWNENABLE	SBDOWNSMODE	CLKCORE1ENABLE	CLKCORE2ENABLE	CLKCORE3ENABLE	MASKPLLGOOD	reserved[1]	reserved[0]	CRGPMPCTRL[3]	CRGPMPCTRL[2]	CRGPMPCTRL[1]	CRGPMPCTRL[0]	DWNNOMCUR[5]	DWNNOMCUR[4]	DWNNOMCUR[3]	DWNNOMCUR[2]	DWNNOMCUR[1]	DWNNOMCUR[0]	DWNPREEMPCUR[5]	DWNPREEMPCUR[4]	DWNPREEMPCUR[3]	DWNPREEMPCUR[2]	DWNPREEMPCUR[1]	DWNPREEMPCUR[0]	TRIGGERCYCLELENGTH[6]	TRIGGERCYCLELENGTH[5]	TRIGGERCYCLELENGTH[4]	TRIGGERCYCLELENGTH[3]	TRIGGERCYCLELENGTH[2]	TRIGGERCYCLELENGTH[1]	TRIGGERCYCLELENGTH[0]	reserved
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Table 39-22. ARH APIX® Configuration Register (ARHn_APCFG01) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31]	DDOWNENABLE	A-Shell Downstream Data Path Configuration '0': Disable data mode / enable pixel stream mode, '1': Enable data mode / disable pixel stream mode Note: For proper operation the following settings are mandatory - PXDATAWIDTH[1:0] = '00' PXINCTRLPIGGYBACK[1:0] = '00'
[30]	SBDOWNSMODE	APIX® PHY Downstream Sideband Data Relation to Core Clock '0': Asynchronous, '1': Synchronous
[29]	CLKCORE1ENABLE	APIX® PHY Core Clock Enable '0': Disable, '1': Enable
[28]	CLKCORE2ENABLE	A-Shell1 Core Clock Enable '0': Disable, '1': Enable

Table 39-22. ARH APIX® Configuration Register (ARHn_APCFG01) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[27]	CLKCORE3ENABLE	A-Shell2 Core Clock Enable '0': Disable, '1': Enable
[26]	MASKPLLGOOD	PLLGOOD Mask bit for APIX® PHY '0': No effect, '1': Masks the actual value of 'pll_good', i.e. 'pll_good' is supposed to be steadily active
[25:24]	reserved	Reserved
[23:20]	CRGPMPCTRL	APIX® PHY Charge Pump Current Control '0000': 5uA '0001': 10uA '0010': 15uA ..... '1111': 80uA
[19:14]	DWNNOMCUR	APIX® Nominal Current Configuration(64 Steps) '000000': Min (0mA - Power down output driver), '111111': Max
[13:8]	DWNPREEMPCUR	APIX® Pre-emphasis Current Configuration(64 Steps) '000000': Min (0mA - Power down output driver), '111111': Max
[7:1]	TRIGGERCYCLELENGTH	A-Shell Sideband Downstream Trigger Output Pulse Pattern Cycle Length Configuration 9: Required when operating in full and half bandwidth mode 18: Required when operating in low bandwidth mode 2 (APIX® PHY core_clk == 62.5 MHz) 36: Required when operating in low bandwidth mode 1 (APIX® PHY core_clk == 62.5 MHz) 36: Required when operating in low bandwidth mode 2 (APIX® PHY core_clk == 125 MHz) 72: Required when operating in low bandwidth mode 1 (APIX® PHY core_clk == 125 MHz)
[0]	reserved	Reserved

39.2.21 ARH APIX® Configuration Register (ARHn_APCFG11 - Channel 1)

The software can use this register to program the APIX® PHY. This register can be written when ARHn_CHCTRL1:TXCFG = '1'.

ARH APIX® Configuration Register (ARHn_APCFG11) Channel 1

Figure 39-26. ARH APIX® Configuration Register (ARHn_APCFG11) Channel 1

ARHn_APCFG11																													
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**
DDOWNENABLE	SBDOWNSMODE	CLKCORE1ENABLE	CLKCORE2ENABLE	CLKCORE3ENABLE	MASKPLLGOOD	reserved	reserved	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	unused	TRIGGERCYCLELENGTH[6]	TRIGGERCYCLELENGTH[5]	TRIGGERCYCLELENGTH[4]	TRIGGERCYCLELENGTH[3]	TRIGGERCYCLELENGTH[2]	TRIGGERCYCLELENGTH[1]	TRIGGERCYCLELENGTH[0]	reserved
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Table 39-23. ARH APIX® Configuration Register (ARHn_APCFG11) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[31]	DDOWNENABLE	A-Shell Downstream Data Path Configuration '0': Disable data mode / enable pixel stream mode, '1': Enable data mode / disable pixel stream mode **Note:** For proper operation the following settings are mandatory: ▫ ARHn_APCFG10:PXDATAWIDTH[1:0] := '00' ▫ ARHn_APCFG10:PXINCTRLPIGGYBACK[1:0] := '00'
[30]	SBDOWNSMODE	APIX® PHY Downstream Sideband Data Relation to Core Clock '0': Asynchronous, '1': Synchronous
[29]	CLKCORE1ENABLE	APIX® PHY Core Clock Enable '0': Disable, '1': Enable
[28]	CLKCORE2ENABLE	A-Shell1 Core Clock Enable '0': Disable, '1': Enable

Table 39-23. ARH APIX® Configuration Register (ARHn_APCFG11) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[27]	CLKCORE3ENABLE	A-Shell2 Core Clock Enable '0': Disable, '1': Enable
[26]	MASKPLLGOOD	PLLGOOD Mask bit for APIX® PHY '0': No effect, '1': Masks the actual value of 'pll_good',
[25:24]	reserved	Reserved
[23:8]	unused	Unused
[7:1]	TRIGGERCYCLELENGTH	A-Shell Sideband Downstream Trigger Output Pulse Pattern Cycle Length Configuration 9: Required when operating in full and half bandwidth mode 18: Required when operating in low bandwidth mode 2 (APIX® PHY core_clk == 62.5 MHz) 36: Required when operating in low bandwidth mode 1 (APIX® PHY core_clk == 62.5 MHz) 36: Required when operating in low bandwidth mode 2 (APIX® PHY core_clk == 125 MHz) 72: Required when operating in low bandwidth mode 1 (APIX® PHY core_clk == 125 MHz)
[0]	reserved	Reserved

39.2.22 ARH APIX® Configuration Register (ARHn_APCFG02 - Channel 0)

The software can use this register to program the APIX® PHY. This register can be written when ARHn_CHCTRL0_TXCFG = '1'.

ARH APIX® Configuration Register (ARHn_APCFG02) Channel 0

Figure 39-27. ARH APIX® Configuration Register (ARHn_APCFG02) Channel 0

ARHn_APCFG02																															
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**
TRIGGERACTIVELENGTH[6]	TRIGGERACTIVELENGTH[5]	TRIGGERACTIVELENGTH[4]	TRIGGERACTIVELENGTH[3]	TRIGGERACTIVELENGTH[2]	TRIGGERACTIVELENGTH[1]	TRIGGERACTIVELENGTH[0]	reserved	TRIGGEROFFSET[6]	TRIGGEROFFSET[5]	TRIGGEROFFSET[4]	TRIGGEROFFSET[3]	TRIGGEROFFSET[2]	TRIGGEROFFSET[1]	TRIGGEROFFSET[0]	reserved	SBUPVALIDACTIVELENGTH[1]	SBUPVALIDACTIVELENGTH[0]	reserved[5]	reserved[4]	reserved[3]	reserved[2]	reserved[1]	reserved[0]	PLLMULT	OSCFILTER	TXINVERT	RXINVERT	PLLGOODSEL	read0	read0	SCPREEN
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-24. ARH APIX® Configuration Register (ARHn_APCFG02) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31:25]	TRIGGERACTIVELENGTH	A-Shell Sideband Downstream Trigger Output Active Length Configuration This bit configures high pulse width of sideband downstream trigger output (in multiples of core clk cycle). 0: 1 cycle 1: 2 cycles 2: 3 cycles .... 71: 72 cycles
[24]	reserved	Reserved

Table 39-24. ARH APIX® Configuration Register (ARHn_APCFG02) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[23:17]	TRIGGEROFFSET	A-Shell Sideband Downstream Trigger Output Offset Configuration This bit configures the start position of sideband downstream trigger output (in multiples of core clk cycle relative to strobe position). 0: 0 cycles 1: 1 cycle 2: 2 cycles .... 71: 71 cycles
[16]	reserved	Reserved
[15:14]	SBUPVALIDACTIVELENGTH	A-Shell Sideband Upstream Valid Output Active Length Configuration This bit configures high pulse width of sideband upstream valid output (in multiples of core clk cycle). '00': 1 cycle '01': 2 cycles '10': 3 cycles '11': 4 cycles
[13:8]	reserved	Reserved
[7]	PLLMULT	APIX® PHY PLL Multiplier Multiplier factor for PLL PLL uses the following reference frequencies: '0': 20MHz '1': 25MHz,
[6]	OSCFILTER	APIX® PHY Oscillator Filter Enable '0': Disable '1': Enable 100MHz LPF in oscillator path,
[5]	TXINVERT	APIX® PHY TX Data Invert Enable '0': Data not inverted '1': Invert data on Tx,
[4]	RXINVERT	APIX® PHY RX Data Invert Enable '0': Data not inverted '1': Invert data from Rx,
[3]	PLLGOODSEL	APIX® PLLGOOD Status Select '0': APIX® PLLGOOD is a reset criteria '1': APIX® PLLGOOD is not a reset criteria,
[2:1]	read0	-

Table 39-24. ARH APIX® Configuration Register (ARHn_APCFG02) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[0]	SCPREEN	Sysclk Prescalar Enable '0': A-Shell clock is equal to bus clock '1': A-Shell clock is half of bus clock,

Sysclk prescalar enable is common across Ashell 0 and 1.

39.2.23 ARH APIX® Configuration Register (ARHn_APCFG12 - Channel 1)

The software can use this register to program the APIX® PHY. This register can be written when ARHn_CHCTRL1_TXCFG = '1'.

ARH APIX® Configuration Register (ARHn_APCFG12) Channel 1

Figure 39-28. ARH APIX® Configuration Register (ARHn_APCFG12) Channel 1

ARHn_APCFG12																															
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**
TRIGGERACTIVELENGTH[6]	TRIGGERACTIVELENGTH[5]	TRIGGERACTIVELENGTH[4]	TRIGGERACTIVELENGTH[3]	TRIGGERACTIVELENGTH[2]	TRIGGERACTIVELENGTH[1]	TRIGGERACTIVELENGTH[0]	reserved	TRIGGEROFFSET[6]	TRIGGEROFFSET[5]	TRIGGEROFFSET[4]	TRIGGEROFFSET[3]	TRIGGEROFFSET[2]	TRIGGEROFFSET[1]	TRIGGEROFFSET[0]	reserved	SBUPVALIDACTIVELENGTH[1]	SBUPVALIDACTIVELENGTH[0]	reserved[5]	reserved[4]	reserved[3]	reserved[2]	reserved[1]	reserved[0]	read0	read0	read0	read0	read0	read0	read0	read0
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-25. ARH APIX® Configuration Register (ARHn_APCFG12) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[31:25]	TRIGGERACTIVELENGTH	A-Shell Sideband Downstream Trigger Output Active Length Configuration This bit configures high pulse width of sideband downstream trigger output (in multiples of core clk cycle). 0: 1 cycle 1: 2 cycles 2: 3 cycles ... 71: 72 cycles
[24]	reserved	Reserved

Table 39-25. ARH APIX® Configuration Register (ARHn_APCFG12) Channel 1 bits

Bit Position	Bit Field Name	Bit Description
[23:17]	TRIGGEROFFSET	A-Shell Sideband Downstream Trigger Output Offset Configuration This bit configures the start position of sideband downstream trigger output (in multiples of core clk cycle relative to strobe position). 0: 0 cycles 1: 1 cycle 2: 2 cycles ... 71: 71 cycles
[16]	reserved	Resernnved
[15:14]	SBUPVALIDACTIVELENGTH	A-Shell Sideband Upstream Valid Output Active Length Configuration This bit configures high pulse width of sideband upstream valid output (in multiples of core clk cycle). '00': 1 cycle '01': 2 cycles '10': 3 cycles '11': 4 cycles
[13:8]	reserved	Reserved
[7:0]	read0	-

39.2.24 ARH APIX® Configuration Register (ARHn_APCFG03 - Channel 0, ARHn_APCFG13 - Channel 1)

The software can use this register to program the A-Shell. This register can be written when ARHn_CHCTRL0:TXCFG = '1'. Register ARHn_APCFG13 (used for channel 1) can be written when ARHn_CHCTRL1:TXCFG = '1'. Register ARHn_APCFG03 for channel 0 is explained here.

ARH APIX® Configuration Register (ARHn_APCFG03) Channel 0

Figure 39-29. ARH APIX® Configuration Register (ARHn_APCFG03) Channel 0

ARHn_APCFG03																															
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**
SBDWNCLK	SBDWNDACLKCLENL[6]	SBDWNDACLKCLENL[5]	SBDWNDACLKCLENL[4]	SBDWNDACLKCLENL[3]	SBDWNDACLKCLENL[2]	SBDWNDACLKCLENL[1]	SBDWNDACLKCLENL[0]	SBUPDWIDTH	SBUPDACLK	SBDWNDWIDTH	SBDWNDACLK[1]	SBDWNDACLK[0]	EPHY	ESHELL	SPIOVERSB	CRCTIMEOUTVAL[3]	CRCTIMEOUTVAL[2]	CRCTIMEOUTVAL[1]	CRCTIMEOUTVAL[0]	WINDOWSIZE[3]	WINDOWSIZE[2]	WINDOWSIZE[1]	WINDOWSIZE[0]	ARQOFF	SUPPRESSITA	reserved[2]	reserved[1]	reserved[0]	SBDWNDACLKCLENU[2]	SBDWNDACLKCLENU[1]	SBDWNDACLKCLENU[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	1	0	0	1	1	0	1	0	1	0	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0

Table 39-26. ARH APIX® Configuration Register (ARHn_APCFG03) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31]	SBDWNCLK	Functional Meaning of Sideband Downstream Trigger This bit defines the functional meaning of sideband downstream trigger when its start position is set to '0'. '0': Request '1': Strobe
[30:24]	SBDWNDACLKCLENL	Sideband Downstream Clock Cycle Time These bits [ARHn_APCFG03:SBDWNDACLKCLENU and SBDWNDACLKCLENL] configure cycle time of sideband downstream clock (in multiples of A-Shell core clock) when sideband downstream data are asynchronous of ARHn_APCFG03:SPIOVERSB is enabled These are lower bits of SBDWNDACLKCLEN[9:0].
[23]	SBUPDWIDTH	Sideband Upstream Ports Enable '0': Enables Sideband upstream data[0] '1': Enables Sideband upstream data[1:0]

Table 39-26. ARH APIX® Configuration Register (ARHn_APCFG03) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[22]	SBUPDACLK	Sideband Upstream Data Validation '0': Sideband upstream data is validated with sideband upstream data valid '1': Sideband upstream data is validated with sideband upstream data[1]
[21]	SBDWNDWIDTH	Sideband Downstream Ports Enable '0': Enables Sideband downstream data[0] '1': Enables Sideband downstream data[1:0]
[20:19]	SBDWNDACLK	Sideband Downstream Clock Generate Source These bits decide the sideband downstream clock generation source. '00': Disable '01': With use of sideband downstream trigger (synchronous to core clk of APIX® PHY) '10': With use of internal counter (asynchronous to core clk of APIX® PHY) '00': Disable
[18]	EPHY	Internal A-Shell to External APIX® PHY Connection Control This bit controls the connection of internal A-Shell to external APIX® PHY through GPIO interface. '0': Disable '1': Enable
[17]	ESHELL	Internal APIX® PHY to External A-Shell Connection Control This bit controls the connection of internal APIX® PHY to external A- Shell through GPIO interface. '0': Disable '1': Enable
[16]	SPIOVERSB	SPI Over Sideband Control '0': SPI transmission over sideband is disabled '1': SPI transmission over sideband is enabled
[15:12]	CRCTIMEOUTVAL	CRC Timeout Value CRC timeout error is generated after N consecutive CRC errors (N = base x multiplier) CRCTIMEOUTVAL[3:2] (Multiplier): '00':1, '01':4, '10':16, '11':128 CRCTIMEOUTVAL[1:0] (Base): '00': 2, '01': 4, '10': 6, '11': 10

Table 39-26. ARH APIX® Configuration Register (ARHn_APCFG03) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[11:8]	WINDOWSIZE	ACKNOWLEDGEMENT WINDOW SIZE These bits defines the window size of the acknowledge protocol (supported sizes:1....12)
[7]	ARQOFF	ARQ Control '0': ARQ enabled '1': ARQ disabled
[6]	SUPPRESSITA	Outbound Idle Transaction Control '0': Outbound Idle transactions are sent '1': Outbound Idle transactions are not sent
[5:3]	reserved	Reserved
[2:0]	SBDWNDACLKCLENU	Sideband Downstream Clock Cycle Time These bits[SBDWNDACLKCLENU and ARHn_APCFG03:SBDWNDACLKCLENL] configure cycle time of sideband downstream clock (in multiples of A-Shell core clock) when sideband downstream data are asynchronous of ARHn_APCFG03:SPIOVERSB is enabled These are upper bits of SBDWNDACLKCLEN[9:0].

39.2.25 ARH Test Register (ARHn_TST)

The software can use this register to test the Event Buffer memory. Write access to this register is protected by PPU attribute.

ARH Test Register (ARHn_TST)

Figure 39-30. ARH Test Register (ARHn_TST)

ARHn_TST																													
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RW	TM	read0	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-27. ARH Test Register (ARHn_TST) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:10]	read0	-
[9]	RW	RAM mode '0': Read mode, '1': Write mode
[8]	TM	Test Mode '0': FIFO operation (Event Buffer mode), '1': RAM operation (Test mode)
[7]	read0	-
[6:0]	ADDR	Address Address of memory to be read or written. The address range is 0-127.

Note: In RAM operation mode (ARHn_TST:TM == '1'), direct access to internal RAM is allowed for debug purpose. Read and write operation over the RAM is possible with RW bit. The address location for read or write is provided in ADDR[6:0] register bits. The read or write data is available in the registers ARHn_EVBUF0 and ARHn_EVBUF1.

39.2.26 ARH Unlock Register (ARHn_UNLOCK)

The software should use this register to lock (0xA86AB10C) or unlock (0xA86ACCE5) ARHn_CHCTRL0, ARHn_CHCTRL1 and ARHn_EVCTRL registers for write access.

ARH Unlock Register (ARHn_UNLOCK)

Figure 39-31. ARH Unlock Register (ARHn_UNLOCK)

ARHn_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-28. ARH Unlock Register (ARHn_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	ARH Unlock The ARH unlock register protects the ARH module from being modified accidentally by software. The ARH registers ARHn_CHCTRL0, ARHn_CHCTRL1 and ARHn_EVCTRL cannot be written until this register has been written with a specific unlock value (0xA86ACCE5). The read to this register always returns a zero. To lock the ARH again software must write another value specific to lock(0xA86AB10C). Write access without unlocking or writing values other than lock value or unlock value to the above mentioned registers cause protection error. **Note:** This register can not be written by 8-bit or 16-bit write access, any such access causes protection error. **Note:** For more details on LOCK and UNLOCK key refer to the device specific datasheet.

39.2.27 ARH Module ID Register (ARHn_MID)

This is a read-only register with a unique Module Identification Number which identifies the version of the ARH module used in the MCU.

ARH Module ID Register (ARHn_MID)

Figure 39-32. ARH Module ID Register (ARHn_MID)

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

Table 39-29. ARH Module ID Register (ARHn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID The ARH module implemented in the device may vary from device to device. This register identifies the particular version of the hardware used in the device. This register helps in developing software to the hardware version implemented in the device. **Note:** For more details on module ID number refer to device specific datasheet.

39.2.28 ARH APIX® Configuration Register (ARHn_APCFG04 - Channel 0, ARHn_APCFG14 - Channel 1)

These registers are used to configure the data and select the clock source for A-Shell. Register ARHn_APCFG04 can be written when ARHn_CHCTRL0:TXCFG = '1'. Register ARHn_APCFG14 (used for channel 1) can be written when ARHn_CHCTRL1:TXCFG = '1'. Register ARHn_APCFG04 for channel 0 is explained here.

ARH APIX ® Configuration Register (ARHn_APCFG04) Channel 0

Figure 39-33. ARH APIX® Configuration Register (ARHn_APCFG04) Channel 0

ARHn_APCFG04																															
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**
SYSCLKSEL	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	AACCLKDATAOFST[1]	AACCLKDATAOFST[0]	AACCLKFREQ[11]	AACCLKFREQ[10]	AACCLKFREQ[9]	AACCLKFREQ[8]	AACCLKFREQ[7]	AACCLKFREQ[6]	AACCLKFREQ[5]	AACCLKFREQ[4]	AACCLKFREQ[3]	AACCLKFREQ[2]	AACCLKFREQ[1]	AACCLKFREQ[0]	DATAPATHSEL	AACCLKMODE
RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-30. ARH APIX® Configuration Register (ARHn_APCFG04) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[31]	SYSCLKSEL	A-Shell0 System Clock Source Selector This bit selects the clock source for APIX®. '0': Clock of 62.5 MHz from APIX® PHY is selected '1': Clock of either same or half of the bus clock frequency (depending on ARHn_APCFG02:SCPREEN) is selected:
[30:24]	read0	-
[23:16]	read0	-

Table 39-30. ARH APIX® Configuration Register (ARHn_APCFG04) Channel 0 bits

Bit Position	Bit Field Name	Bit Description
[15:14]	AACCLKDATAOFST	AIC2 Clock Data Offset These bits configures offset between the edges of AIC2 clock and AIC2 data in multiples of core clock '00': 1 core clock cycle '01': 2 core clock cycles '10': 3 core clock cycles '11': 4 core clock cycles **Note:** The core clock refers to the system clock selected by ARHn_APCFG04:SYSCLKSEL
[13:2]	AACCLKFREQ	AIC2 Clock Frequency 0x00000000: AIC2 Clock generator is off 0x00000001: This configuration is not allowed 0x00000002-0xFFFFFFFF -> N(2.....4095) AIC2 Clock Frequency = core clock/(2*N) If ARHn_APCFG04:AACCLKMODE=0: A-Shell0 bandwidth gross = AACCLKFREQ * 2 * AAC_data_width If ARHn_APCFG04:AACCLKMODE=1: A-Shell0 bandwidth gross = AACCLKFREQ * 1 * AAC_data_width **Notes:** 1. AAC_data_width = {1,2} 2. The core clock refers to the system clock selected by ARHn_APCFG04:SYSCLKSEL
[1]	DATAPATHSEL	Data Path Selection '0' : A-Shell is connected to APIX® PHY '1' : A-Shell is connected to AIC2:
[0]	AACCLKMODE	AIC2 Clock Mode '0': Data transfers at both edges of AIC2 clock '1': Data transfers at rising edges of AIC2 clock only:

39.2.29 APIX® PHY Evaluation Transmitter Pattern Register (ARHn_EVAL0)

This register is used to configure the transmitter pattern for APIX® PHY evaluation. Write access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Transmitter Pattern Register (ARHn_EVAL0)

Figure 39-34. APIX® PHY Evaluation Transmitter Pattern Register (ARHn_EVAL0)

ARHn_EVAL0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	TXPATTERN[15]	TXPATTERN[14]	TXPATTERN[13]	TXPATTERN[12]	TXPATTERN[11]	TXPATTERN[10]	TXPATTERN[9]	TXPATTERN[8]	TXPATTERN[7]	TXPATTERN[6]	TXPATTERN[5]	TXPATTERN[4]	TXPATTERN[3]	TXPATTERN[2]	TXPATTERN[1]	TXPATTERN[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-31. APIX® PHY Evaluation Transmitter Pattern Register (ARHn_EVAL0) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	TXPATTERN	TX Pattern Configuration Configures the 16-bit user defined pattern to be used in the APIX¬Æ PHY pattern generator.

39.2.30 APIX® PHY Evaluation Receiver Pattern Register (ARHn_EVAL1)

This register shows the receiver pattern of APIX® PHY evaluation. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Receiver Pattern Register (ARHn_EVAL1)

Figure 39-35. APIX® PHY Evaluation Receiver Pattern Register (ARHn_EVAL1)

ARHn_EVAL1																															
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
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RXPATTERN[15]	RXPATTERN[14]	RXPATTERN[13]	RXPATTERN[12]	RXPATTERN[11]	RXPATTERN[10]	RXPATTERN[9]	RXPATTERN[8]	RXPATTERN[7]	RXPATTERN[6]	RXPATTERN[5]	RXPATTERN[4]	RXPATTERN[3]	RXPATTERN[2]	RXPATTERN[1]	RXPATTERN[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-32. APIX® PHY Evaluation Receiver Pattern Register (ARHn_EVAL1) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	RXPATTERN	RX Pattern Status 16-bit pattern received at RX (aligns when preamble is selected by ARHn_EVAL2:PATTERNGENEN[1:0].

39.2.31 APIX® PHY Evaluation Configuration Register (ARHn_EVAL2)

This register is used to configure the APIX® PHY for evaluation. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Configuration Register (ARHn_EVAL2)

Figure 39-36. APIX® PHY Evaluation Configuration Register (ARHn_EVAL2)

ARHn_EVAL2																															
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
PATTERNGENEN[1]	PATTERNGENEN[0]	RXEDGE	PATTERNRATE	PATTCHKSOURCE	ERRORHOLD[1]	ERRORHOLD[0]	read0	TXSOURCE[2]	TXSOURCE[1]	TXSOURCE[0]	ENLOOPBCK[1]	ENLOOPBCK[0]	RXEPTRIGGER	read0	read0	LOOPBCKSWING[3]	LOOPBCKSWING[2]	LOOPBCKSWING[1]	LOOPBCKSWING[0]	read0	read0	read0	read0	TESTPD	read0	read0	MASKPLLGOODSET	MASKPLLGOODCLR	RXREALIGNSET	RXREALIGNCLR	TESTENABLE
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	RpWp	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

Table 39-33. APIX® PHY Evaluation Configuration Register (ARHn_EVAL2) bits

Bit Position	Bit Field Name	Bit Description
[31:30]	PATTERNGENEN	Pattern Generated and/or Checked '00': 16 bit user defined pattern, inverting after 16bits '01': 16 bit user defined pattern, '10': 10-bit PRBS '11': Preamble 55AA (Pattern checker aligns to word. This is independent of TxSource)
[29]	RXEDGE	Edge Selector for Sampling '0': Sample on rising edge of 500MHz '1': Sample on falling edge of 500MHz,
[28]	PATTERNRATE	Pattern Rate Selector(for generator and checker) '0': According to ARHn_APCFG00:TXBWMODE '1': 62.5Mb/s (for upstream test)
[27]	PATTCHKSOURCE	Pattern Check Source Select source of data for pattern checker. '0': From samplers '1': RxBit

Table 39-33. APIX® PHY Evaluation Configuration Register (ARHn_EVAL2) bits

Bit Position	Bit Field Name	Bit Description
[26:25]	ERRORHOLD	Hold Time for ARHn_EVAL2:RXEQTX on Error '0': 16ns (1 cycle period of 62.5MHz clock) '1': 256ns (16 cycle periods of 62.5MHz clock) 2: Until next ARHn_EVAL3:RXTXLOCKED 3: Until ARHn_EVAL2:PATTERNGENEN set to preamble
[24]	read0	-
[23:21]	TXSOURCE	Transmit Data Source '000': TxData from core (normal operation) '001': Pattern generator '010': Sampled data (500Mbps) '011': RxBit '100': 500Mbps Toggle (01010101) '101': 250Mbps Toggle '110': 125Mbps Toggle '111': 62.5Mbps Toggle, The pattern generator is independent of TXSOURCE.
[20:19]	ENLOOPBCK	Enable Loopback Loopback from TX output to RX input '0x': Loopback disabled, '11': Loopback from SDOUT '10': Loopback from serializer output
[18]	RXEPTRIGGER	ARHn_EVAL3:RXEDGEPOS[3] Enable 0->1 Sets ARHn_EVAL3:RXEDGEPOS[3] so that ARHn_EVAL3:RXEDGEPOS is in range 4..11 After RXEPTRIGGER, ARHn_EVAL3:RXEDGEPOS indicates a shift of minimum 4500MHz clock cycles (=0.5 UI @ 62.5Mbps) before wrapping.
[17:16]	read0	-
[15:12]	LOOPBCKSWING	Loopback Swing Control '0000': 48mV (differential) '1111': 192mV (differential),
[11:8]	read0	-

Table 39-33. APIX® PHY Evaluation Configuration Register (ARHn_EVAL2) bits

Bit Position	Bit Field Name	Bit Description
[7]	TESTPD	Power down for Test This bit powers down complete APIX¬Æ PHY macro. Used for production test only. '0': No effect, '1': Power OFF. Overrides ARHn_APCFG00:ENPLL, ARHn_APCFG00:OFFSETCOMP, ARHn_APCFG00:ENDOWNSTREAM and ARHn_APCFG00:ENUPSTREAM power control signals.
[6:5]	read0	-
[4]	MASKPLLGOODSET	A-Shell0 MASKPLLGOOD Set This bit sets the MASKPLLGOOD output of A-Shell0. '0': No effect, '1': Sets the MASKPLLGOOD output of A-Shell0 **Note:** that ARHn_EVAL2:MASKPLLGOODCLR has got higher priority than MASKPLLGOODSET.
[3]	MASKPLLGOODCLR	A-Shell0 MASKPLLGOOD Clear This bit clears MASKPLLGOOD output of A-Shell0. '0': Clears the MASKPLLGOOD output of A-Shell0 '1': No effect,
[2]	RXREALIGNSET	A-Shell0 RXREALIGN Set Bit '0': No effect '1': Sets the RXREALIGN output from A-Shell0 Note that ARHn_EVAL2:RXREALIGNCLR has got higher priority than RXREALIGNSET.
[1]	RXREALIGNCLR	A-Shell0 RXREALIGN Clear Bit '0': Clears the RXREALIGN output of A-Shell0 '1': No change,
[0]	TESTENABLE	Enable BIST '0': Normal operation '1': Test using pattern generator and checker (BIST controls ARHn_EVAL2:RXEDGE and ARHn_EVAL2:PATTERNGENEN) 0 ->1 Edge triggers start of test 1 ->0 Edge interrupts any running test

39.2.32 APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3)

This register gives the status of various attributes of receiver. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3)

Figure 39-37. APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3)

ARHn_EVAL3																															
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**
RXTXLOCKED	RXEQTX	read0	read0	RXEDGEPOS[3]	RXEDGEPOS[2]	RXEDGEPOS[1]	RXEDGEPOS[0]	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0
Rp	Rp	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-34. APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3) bits

Bit Position	Bit Field Name	Bit Description
[31]	RXTXLOCKED	Preamble Lock Status '0': Preamble expected, but not yet detected '1': Preamble pattern has been detected and pattern checker has aligned to preamble word. Checker expects to receive pattern (or PRBS) data (starting with first word that is not preamble)
[30]	RXEQTX	RX Pattern status '0': RX pattern is not detected (because of Error/LoopBack dis- abled/Preamble not yet detected) '1': RX pattern is as expected from TX pattern generator (goes high when preamble is detected)
[29:28]	read0	-

Table 39-34. APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL3) bits

Bit Position	Bit Field Name	Bit Description
[27:24]	RXEDGEPOS	Position of Last Edge in Upstream Data bits[2:0] 0: Data edge is aligned to 62.5MHz clock rising edge 1..7: Number of 500MHz clocks that data edge lags rising edge of 62.5MHz clock bit[3] Toggles state when RXEDGEPOS[2:0] crosses 000/111 boundary Sets to !RXEDGEPOS[2] at rising edge of ARHn_EVAL2:RXEPTRIGGER
[23:0]	read0	-

39.2.33 APIX® PHY Evaluation Misc Test Enable Register (ARHn_EVAL4)

This register is used to enable various miscellaneous test functions and to program length of the BIST pattern for APIX® PHY. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Misc Test Enable Register (ARHn_EVAL4)

Figure 39-38. APIX® PHY Evaluation Receiver Status Register (ARHn_EVAL4)

ARHn_EVAL4																															
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**
TESTMISC[25]	TESTMISC[24]	TESTMISC[23]	TESTMISC[22]	TESTMISC[21]	TESTMISC[20]	TESTMISC[19]	TESTMISC[18]	TESTMISC[17]	TESTMISC[16]	TESTMISC[15]	TESTMISC[14]	TESTMISC[13]	TESTMISC[12]	TESTMISC[11]	TESTMISC[10]	TESTMISC[9]	TESTMISC[8]	TESTMISC[7]	TESTMISC[6]	TESTMISC[5]	TESTMISC[4]	TESTMISC[3]	TESTMISC[2]	TESTMISC[1]	TESTMISC[0]	LOCK2PATTERN	TESTLENGTH[4]	TESTLENGTH[3]	TESTLENGTH[2]	TESTLENGTH[1]	TESTLENGTH[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Table 39-35. APIX® PHY Evaluation Misc Test Enable Register (ARHn_EVAL4) bits

Bit Position	Bit Field Name	Bit Description
[31:6]	TESTMISC	Test Functions Selection
[5]	LOCK2PATTERN	Lock Pattern Selection '0': Lock to preamble {55AA55AA} '1': Skip preamble, lock to pattern
[4:0]	TESTLENGTH	Length of BIST Test Pattern (excluding preamble) 0-29: (2^TestLength) clocks 30-31: No pattern test (preamble only),

39.2.34 APIX® PHY Evaluation Status Register (ARHn_EVAL5)

This register gives the status of the tests performed on APIXR PHY. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Status Register (ARHn_EVAL5)

Figure 39-39. APIX® PHY Evaluation Status Register (ARHn_EVAL5)

ARHn_EVAL5																															
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**
TESTSTATUS[31]	TESTSTATUS[30]	TESTSTATUS[29]	TESTSTATUS[28]	TESTSTATUS[27]	TESTSTATUS[26]	TESTSTATUS[25]	TESTSTATUS[24]	TESTSTATUS[23]	TESTSTATUS[22]	TESTSTATUS[21]	TESTSTATUS[10]	TESTSTATUS[19]	TESTSTATUS[18]	TESTSTATUS[17]	TESTSTATUS[16]	TESTSTATUS[15]	TESTSTATUS[14]	TESTSTATUS[13]	TESTSTATUS[12]	TESTSTATUS[11]	TESTSTATUS[10]	TESTSTATUS[9]	TESTSTATUS[8]	TESTSTATUS[7]	TESTSTATUS[6]	TESTSTATUS[5]	TESTSTATUS[4]	TESTSTATUS[3]	TESTSTATUS[2]	TESTSTATUS[1]	TESTSTATUS[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-36. APIX® PHY Evaluation Status Register (ARHn_EVAL5) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	TESTSTATUS	Status of BIST [0]: Edge Selector for Sampling from BIST [1]: BIST running '0' = Not started or complete '1' = Running (or not completed at TestEnable low): [2]: BIST pass '0' = Fail or running '1' = Pass: [3]: Pattern type at test end (pass or fail) '0' = Preamble '1' = Pattern: [4]: Current pattern type '0' = Preamble '1' = Pattern: [31:5] Reserved:

39.2.35 APIX® PHY Evaluation Misc Test Configuration Register (ARHn_EVAL6)

This register is used to configure for various miscellaneous test functions for APIX® PHY. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Misc Test Configuration Register (ARHn_EVAL6)

Figure 39-40. APIX® PHY Evaluation Misc Test Configuration Register (ARHn_EVAL6)

ARHn_EVAL6																															
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**
AXMCFG[31]	AXMCFG[30]	AXMCFG[29]	AXMCFG[28]	AXMCFG[27]	AXMCFG[26]	AXMCFG[25]	AXMCFG[24]	AXMCFG[23]	AXMCFG[22]	AXMCFG[21]	AXMCFG[20]	AXMCFG[19]	AXMCFG[18]	AXMCFG[17]	AXMCFG[16]	AXMCFG[15]	AXMCFG[14]	AXMCFG[13]	AXMCFG[12]	AXMCFG[11]	AXMCFG[10]	AXMCFG[9]	AXMCFG[8]	AXMCFG[7]	AXMCFG[6]	AXMCFG[5]	AXMCFG[4]	AXMCFG[3]	AXMCFG[2]	AXMCFG[1]	AXMCFG[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-37. APIX® PHY Evaluation Misc Test Configuration Register (ARHn_EVAL6) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	AXMCFG	Misc Test Configuration Configuration for Misc Test (details unavailable).

39.2.36 APIX® PHY Evaluation Misc Test Status Register (ARHn_EVAL7)

This register gives the status of various miscellaneous tests for APIX® PHY. Access to this register is protected by PPU test attribute.

APIX® PHY Evaluation Misc Test Status Register (ARHn_EVAL7)

Figure 39-41. APIX® PHY Evaluation Misc Test Status Register (ARHn_EVAL7)

ARHn_EVAL6																															
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**
AXMST[31]	AXMST[30]	AXMST[29]	AXMST[28]	AXMST[27]	AXMST[26]	AXMST[25]	AXMST[24]	AXMST[23]	AXMST[22]	AXMST[21]	AXMST[20]	AXMST[19]	AXMST[18]	AXMST[17]	AXMST[16]	AXMST[15]	AXMST[14]	AXMST[13]	AXMST[12]	AXMST[11]	AXMST[10]	AXMST[9]	AXMST[8]	AXMST[7]	AXMST[6]	AXMST[5]	AXMST[4]	AXMST[3]	AXMST[2]	AXMST[1]	AXMST[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 39-38. APIX® PHY Evaluation Misc Test Status Register (ARHn_EVAL7) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	AXMST	Misc Test Status Status of Misc Tests (details unavailable).

39.3 Operation of ARH

This chapter describes the operation of ARH.

39.3.1 Data transfers (frame types)

The various frame/message types generated and handled by ARH are shown in this section.

Downstream write transaction (Remote Handler TX -> Remote Handler RX)

Table 39-39. Write message transmitted by ARH to remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
55	0	Control[3:0]	Tsize	1		Transaction size: '00' = byte, '01' = half word (2 bytes), '10' = word (4 bytes), '11' = Reserved
54			Tsize	0		
53			OAen			Offset address enable
52			WRn		1	Write transaction
51		Addr[19:0]	Addr	19		Byte address
50			Addr	18		
49			Addr	17		
48			Addr	16		
47	1		Addr	15		
46			Addr	14		
45			Addr	13		
44			Addr	12		
43			Addr	11		
42			Addr	10		
41			Addr	9		
40			Addr	8		
39	2		Addr	7		
38			Addr	6		
37			Addr	5		
36			Addr	4		
35			Addr	3		
34			Addr	2		
33			Addr	1		
32			Addr	0		

Table 39-39. Write message transmitted by ARH to remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
31	3	Data[31:0]	Data	31		Data, If Tsize = '00': then Data[31:24] is used, If Tsize = '01': then Data[31:16] is used If Tsize = '10': then Data[31:0] is used
30			Data	30		
29			Data	29		
28			Data	28		
27			Data	27		
26			Data	26		
25			Data	25		
24			Data	24		
23	4		Data	23		
22			Data	22		
21			Data	21		
20			Data	20		
19			Data	19		
18			Data	18		
17			Data	17		
16			Data	16		
15	5		Data	15		
14			Data	14		
13			Data	13		
12			Data	12		
11			Data	11		
10			Data	10		
9			Data	9		
8			Data	8		
7	6		Data	7		
6			Data	6		
5			Data	5		
4			Data	4		
3			Data	3		
2			Data	2		
1			Data	1		
0			Data	0		

Downstream read request transaction (From ARH TX -> Remote ARH RX)

Table 39-40. Read request message transmitted by ARH to remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
55	0	Control[3:0]	Tsize	1		Transaction size: '00' = byte, '01' = half word (2 bytes), '10' = word (4 bytes), '11' = Reserved
54			Tsize	0		
53			OAen			Offset address enable
52			WRn		0	Read request transaction
51		Addr[19:0]	Addr	19		Byte address
50			Addr	18		
49			Addr	17		
48			Addr	16		
47	1		Addr	15		
46			Addr	14		
45			Addr	13		
44			Addr	12		
43			Addr	11		
42			Addr	10		
41			Addr	9		
40			Addr	8		
39	2		Addr	7		
38			Addr	6		
37			Addr	5		
36			Addr	4		
35			Addr	3		
34			Addr	2		
33			Addr	1		
32			Addr	0		
31	3	Idx[7:0]	Idx	7		Read index
30			Idx	6		
29			Idx	5		
28			Idx	4		
27			Idx	3		
26			Idx	2		
25			Idx	1		
24			Idx	0		

Table 39-40. Read request message transmitted by ARH to remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
23	4	Reserved	-			Unused
22			-			
21			-			
20			-			
19			-			
18			-			
17			-			
16			-			
15	5		-			
14			-			
13			-			
12			-			
11			-			
10			-			
9			-			
8			-			
7	6		-			
6			-			
5			-			
4			-			
3			-			
2			-			
1			-			
0			-			

Upstream read result (Remote Handler RX -> Remote Handler TX)

Table 39-41. Read response message received by ARH from remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
55	0	Control[3:0]	Tsize	1		Transaction size: '00' = byte, '01' = half-word(2 bytes), '10' = word (4 bytes), '11' = Reserved
54			Tsize	0		
53			OAen		-	Offset address enable
52			WRn		0	Read result transaction
51		Reserved	Err			Chip internal bus error
50			-			Unused
49			-			
48			-			
47	1	Idx[7:0]	Idx	7		Read index
46			Idx	6		
45			Idx	5		
44			Idx	4		
43			Idx	3		
42			Idx	2		
41			Idx	1		
40			Idx	0		
39	2	Reserved	-			Unused
38			-			
37			-			
36			-			
35			-			
34			-			
33			-			
32			-			

Table 39-41. Read response message received by ARH from remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
31	3	Data[31:0]	Data	31		Read result data, If Tsize = '00': then Data[31:24] is used, If Tsize = '01': then Data[31:16] is used, If Tsize = '10': then Data[31:0] is used
30			Data	30		
29			Data	29		
28			Data	28		
27			Data	27		
26			Data	26		
25			Data	25		
24			Data	24		
23	4		Data	23		
22			Data	22		
21			Data	21		
20			Data	20		
19			Data	19		
18			Data	18		
17			Data	17		
16			Data	16		
15	5		Data	15		
14			Data	14		
13			Data	13		
12			Data	12		
11			Data	11		
10			Data	10		
9			Data	9		
8			Data	8		
7	6		Data	7		
6			Data	6		
5			Data	5		
4			Data	4		
3			Data	3		
2			Data	2		
1			Data	1		
0			Data	0		

Upstream event message (Remote Handler RX -> Remote Handler TX)

Table 39-42. Event message received by ARH from remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
55	0	Control[3:0]	Tsize	1	11	Event message received
54			Tsize	0		
53			OAen		-	Offset address enable
52			WRn		-	Read result transaction
51		Reserved	Err			Chip internal bus error
50			-			
49			-			
48			-			
47	1	Idx[7:0]	Idx	7		Event index
46			Idx	6		
45			Idx	5		
44			Idx	4		
43			Idx	3		
42			Idx	2		
41			Idx	1		
40			Idx	0		
39	2	Reserved	-			Reserved
38			-			
37			-			
36			-			
35			-			
34			-			
33			-			
32			-			

Table 39-42. Event message received by ARH from remote (Remote Handler RX)

Bit	Byte	Field	Bits		Value	Description
31	3	Data[31:0]	Data	31		Read result data
30			Data	30		
29			Data	29		
28			Data	28		
27			Data	27		
26			Data	26		
25			Data	25		
24			Data	24		
23	4		Data	23		
22			Data	22		
21			Data	21		
20			Data	20		
19			Data	19		
18			Data	18		
17			Data	17		
16			Data	16		
15	5		Data	15		
14			Data	14		
13			Data	13		
12			Data	12		
11			Data	11		
10			Data	10		
9			Data	9		
8			Data	8		
7	6		Data	7		
6			Data	6		
5			Data	5		
4			Data	4		
3			Data	3		
2			Data	2		
1			Data	1		
0			Data	0		

39.4 Notes on using ARH

This section is the 'programmer's guide', which lists the usage notes for programming the ARH module. It is recommended to read these guidelines before programming the ARH module.

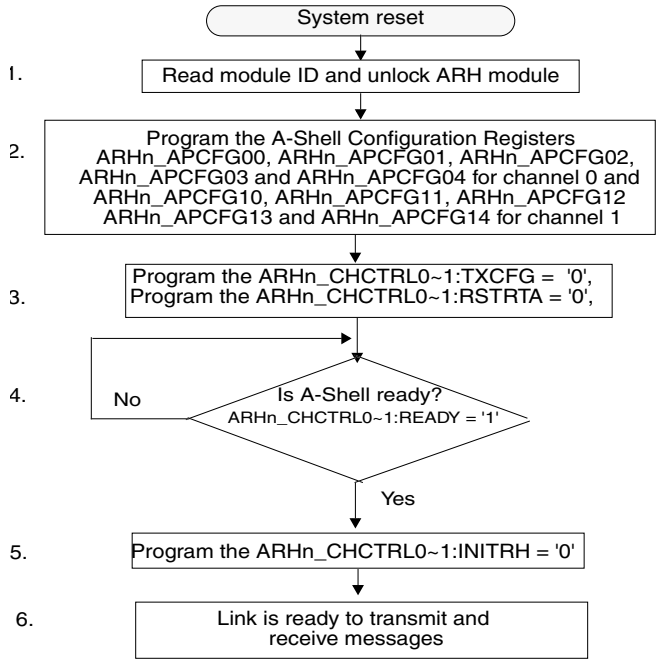
General usage notes

Reserved bits return undefined values. The software programs shall be independent of the values read from the reserved register bits.

Steps in programming the ARH module

Figure 39-42 gives the general steps a programmer shall follow while using the ARH module.

Figure 39-42. Programmer's flowcharts



After the system reset the steps in programming the ARH module are as follows:

1. The software detects the module ID number of ARH by reading the ARHn_MID register. This helps it in identifying the attributes and capabilities supported by the ARH module. The software unlocks the ARH module by writing proper unlocking key to the ARHn_UNLOCK register. After unlocking the module the software configures ARH by setting appropriate registers.
2. The software must program the ARH A-Shell Configuration Registers:
   - ARHn_APCFG00, ARHn_APCFG01, ARHn_APCFG02, ARHn_APCFG03 and ARHn_APCFG04 for channel 0
   - ARHn_APCFG10, ARHn_APCFG11, ARHn_APCFG12, ARHn_APCFG13 and ARHn_APCFG14 for channel 1
3. Program the ARHn_CHCTRL0~1:TXCFG bit to '0', this makes the programming of ARH A- Shell Configuration Registers done in step 2 effective. The software should also program the ARHn_CHCTRL0~1:RSTRTA bit to '0', this restarts the A-Shell.
4. The software can then monitor the A-Shell status bits ARHn_CHCTRL0~1:CONNECTED to see if the channel is established.
5. The software must program the ARHn_CHCTRL0~1:INITRH bit to '0', this will initialize the Remote Handler.
6. The link is ready to transmit and receive messages. The software should now appropriately program the ARH Transaction Buffer Control Registers ARHn_TBCTRL00~15, ARH Transaction Frame Registers ARHn_TFCTRL00, ARHn_TFADDR00~15, ARHn_TFDATA00, and ARH Event Control Registers ARHn_EVCTRL.

Note: For more details to program the A-Shell/APIX® PHY Configuration Registers refer to 39.5 Use cases of this document.

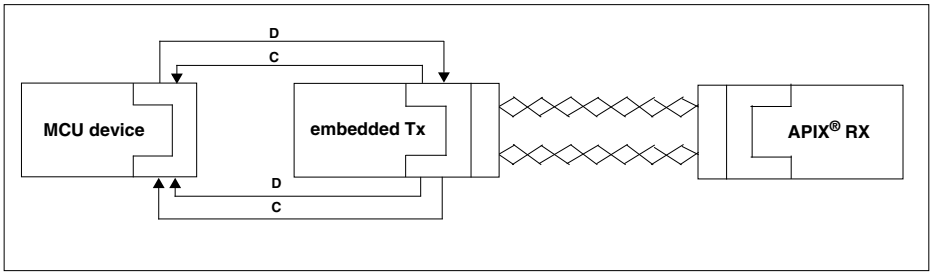
39.5 Use cases

This section list various use cases for communication.

39.5.1 Communication over Automotive Interconnect to external A-Shell

1-bit data width

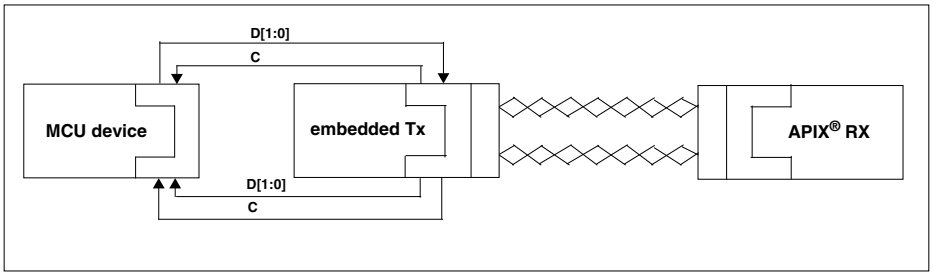
Figure 39-43. 1-bit data width communication over AIC link to external A-Shell



Register	Bit	Default	Value	Description
ARHn_APCFGn1	31	'1'	'0'	'0': Disable data mode/enable pixel stream mode '1': Enable data mode/disable pixel stream mode
ARHn_APCFGn1	29	'1'	'0'	'0': Disable data mode/enable pixel stream mode '1': Enable data mode/disable pixel stream mode
ARHn_APCFGn3	23	'1'	'0'	'0': sbup_data[0] '1': sbup_data[1:0]
ARHn_APCFGn3	21	'1'	'0'	'0': sbdown_data[0] '1': sbdown_data[1:0]
ARHn_APCFGn3	18	'0'	'1'	A-Shell: connect internal A-Shell to external APIX® PHY through GPIO interface '0': Disable '1': Enable

2-bit data width

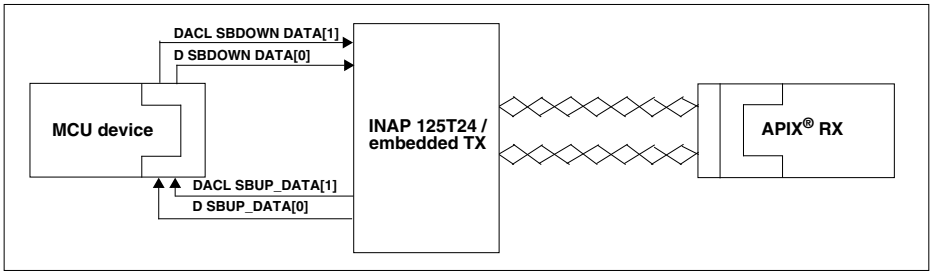
Figure 39-44. 2-bit data width communication over AIC link to external A-Shell



Register	Bit	Default	Value	Description
ARHn_APCFGn1	31	'1'	'0'	'0': Disable data mode/enable pixel stream mode '1': Enable data mode/disable pixel stream mode
ARHn_APCFGn1	29	'1'	'0'	'0': Disable data mode/enable pixel stream mode '1': Enable data mode/disable pixel stream mode
ARHn_APCFGn3	18	'0'	'1'	A-Shell: connect internal A-Shell to external APIX® PHY through GPIO interface '0': Disable '1': Enable

39.5.2 Communication over Automotive Interconnect to external PHY

Figure 39-45. Communication over AIC link to external PHY

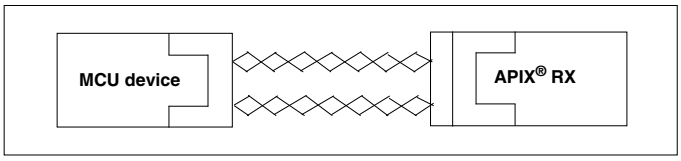


Register	Bit	Default	Value	Description
ARHn_APCFGn1	31	'1'	'0'	'0': Disable data mode/enable pixel stream mode '1': Enable data mode/disable pixel stream mode
ARHn_APCFGn1	29	'1'	'0'	'0': Disable data mode/enable pixel stream mode '1': Enable data mode/disable pixel stream mode
ARHn_APCFGn3	22	'0'	'1'	A-Shell: validate sbup_data with '0': sbup_valid '1': sbup_data[1]
ARHn_APCFGn3	20	'0'	'1'	A-Shell: generate sbdown clock and transmit as sbdown_data[1] '00': Disable '01': With use of sbdown_trigger (synchronous to core_clk of APIX® PHY) '10': With use of internal counter (asynchronous to core_clk of APIX® PHY) '11': Disable
	19	'0'	'0'	
ARHn_APCFGn3	18	'0'	'1'	A-Shell: connect internal A-Shell to external APIX® PHY through GPIO interface '0': Disable '1': Enable
ARHn_APCFGn3	2	'0'	As per requirement	A-Shell: Configures cycle time of sbdown clock (multiples of A-Shell core clock) when sbdown_data are asynchronous (sbdown_data[1] is used as sbdown clock) or cfg_spi_over_sb is enabled. 0x0B: Recommended minimum (no low bandwidth mode, A-Shell and APIX® PHY operate at same core clock frequency) 0x14: Recommended minimum (low bandwidth mode 2, A-Shell, and APIX® PHY operate at 62.5 MHz) 0x26: Recommended minimum (low bandwidth mode 1, A-Shell, and APIX® PHY operate at 62.5 MHz).
	1	'0'		
	0	'0'		
	30	'0'		
	29	'1'		
	28	'0'		
	27	'0'		
	26	'1'		
	25	'1'		
	24	'0'		

Note: If n = 0 indicates channel 0, n = 1 indicates channel 1.

39.5.3 Communication over APIX® link (internal PHY) to Remote RX PHY

Figure 39-46. Communication over APIX® link (internal PHY) to remote RX PHY



Downlink over Pixelchannel

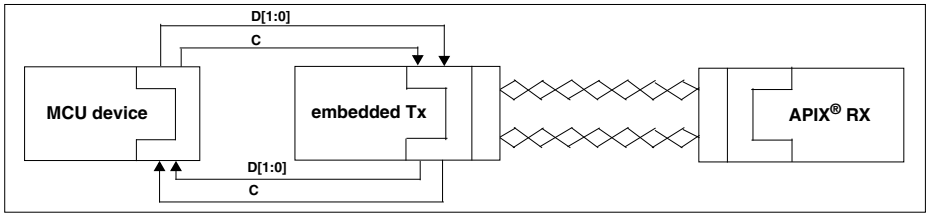
Downlink over Pixelchannel is provided through register ARHn_APCFG03:EPHY with the default value set to '1'.The CY9DF126 (ATLAS) device can be used with baud rates of 125Mbits/s, 250Mbits/s and 500Mbits/s. These rates are set using ARHn_APCFG00:TXBWMODE.

Downlink over Sidebandchannel

Register	Bit	Default	Value	Description
APCFG01	31	1	0	0: Disable data mode/Enable pixel stream mode 1: Enable data mode/Disable pixel stream mode
APCFG03	18	1	0	0: Disable the connection of internal A-Shell to external APIX® PHY through GPIO interface 1: Enable the connection of internal A-Shell to external APIX® PHY through GPIO interface

39.5.4 Communication over Automotive Interconnect-2 to External Automotive Interconnect-2

Figure 39-47. Communication over Automotive Interconnect-2 to External Automotive Interconnect-2



Register	Bit	Default	Value	Description
APCFGn4	1	0	1	0:A-Shell is connected to APIX® PHY 1:A-Shell is connected to AIC2

If n = 0 indicates channel 0, n = 1 indicates channel 1.

Note: All bits of APCFGn4 register have to be programmed, depending on required settings for remote device.

40. High-Performance Matrix

This chapter briefly explains the High-Performance Matrix (HPM).

40.1 Outline of the High-Performance Matrix

This section briefly describes the features and the block diagram of the High-Performance Matrix (HPM).

Features of High-Performance Matrix

The High-Performance Matrix (HPM) is a configured and extended version of the Arm PrimeCell fast Interconnect (PL301). The HPM is a highly configurable Advanced Microcontroller Bus Architecture (AMBA) 3 bus interconnect. The interconnect consists of a central AXI cross-bar switch, known as the AXI bus matrix, and AMBA infrastructure components to enable connection to other AMBA devices.

Features of the HPM include:

- A configurable number of SIs and MIs
- Multi-layer AXI routing, suitable for high-performance applications
- Sparse connection options to reduce gate count
- An arbitration mechanism
- Support for multiple clock domains
- A PrimeCell ID register to aid self-discovery in systems
- A configurable memory map

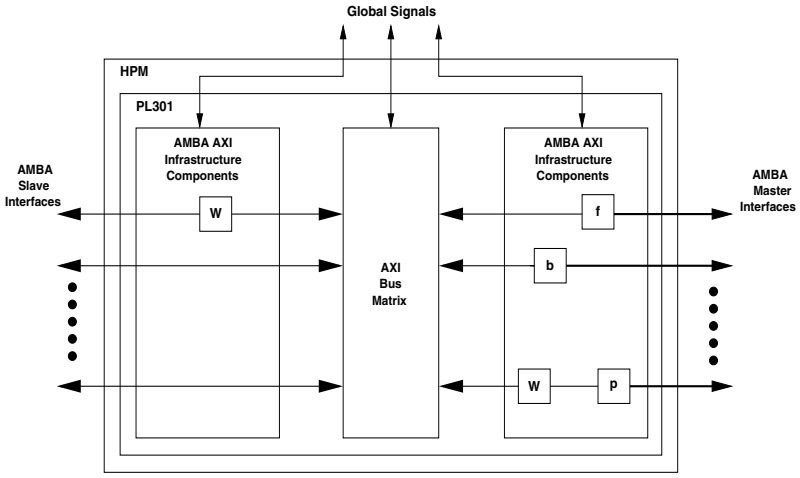
For a detailed list of features refer to the PrimeCell High-Performance Matrix (PL301) Technical Reference Manual and the following link:

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0422d/DDI0422D_hpm_pl301_r1p2_ts.pdf

Block diagram of the HPM

The Figure 40-1 illustrates the top-level hierarchy of an interconnect

Figure 40-1. Block diagram of High Performance Matrix



In Figure 40-1 the AMBA AXI infrastructure components types are as follows:

b - Buffering components

f - Frequency conversion components

p - Data bus protocol conversion components

W - Data bus width conversion components.

This combination of IPs provides support for other AMBA bus protocols including AHB-Lite and Advanced Peripheral Bus (APB™).

Note: For more indepth features refer to the Arm documentation listed in the reference section.

40.2 Arbitration scheme for the HPM

On certain devices the HPM implements the programmable Least Recently Granted (LRG) scheme.

In the Least Recently Granted (LRG) scheme, each connected SI has a single slot associated with it, but each interface also has a priority value. This priority value, whose post-reset value can be configured at design time and programmed or interrogated through the APB programming interface, can make the arbiter behave as a:

- Pure LRG scheme
- Fixed priority encode
- Combination of the two

All masters with the same priority form a priority group. As a result of arbitration, a master can move within its priority group but cannot leave its group. Thus, no new masters can join the group. Arbitration is granted to the highest priority group from which a member is trying to win access, and within that group, to the highest master at that time. When a master wins arbitration, it is relegated to the bottom of its group to ensure that is cannot prevent other masters in its group from accessing the slave.

If you configure all master priorities to different levels, the arbiter implements a fixed priority scheme. This occurs because in this case, each master is in a group of its own, and therefore, masters maintain their ordering.

If all master priorities are the same, then an LRG scheme is implemented. The reason that it behaves as an LRG is because the process of relegating the master that was last granted access, to the bottom of its group, results in the masters being ordered from the LRG master at the top, to the most recently granted at the bottom.

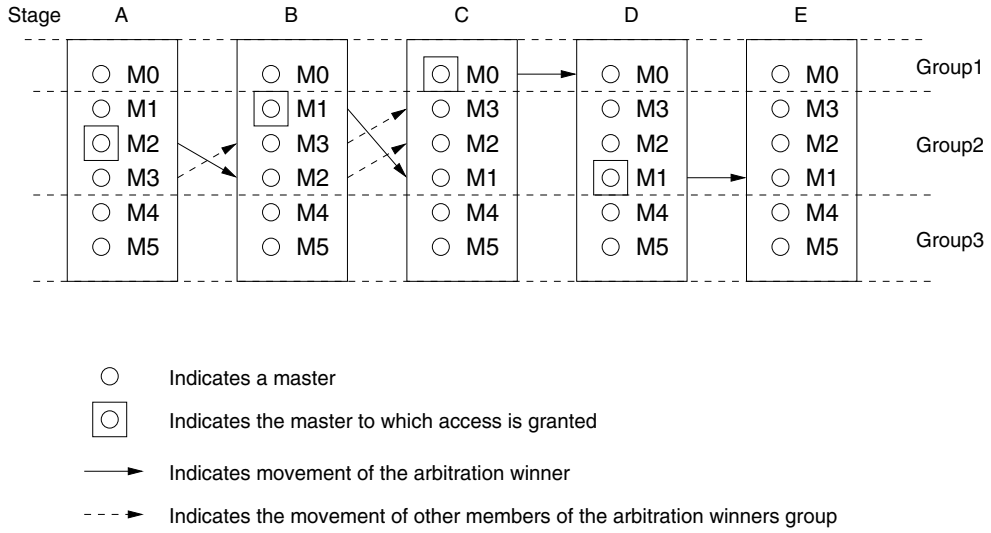
The LRG and fixed priority modes concurrently exist when the master priority value registers are programmed with a combination of identical and unique values. You can mix priority groups that contain one member with priority groups that contain more than one member in an arbitrary manner. The arbiter places no restriction on the number of groups or their membership.

Figure 40-2 shows an example of the LRG arbitration scheme.

For the software configurable options and the reset values of the interfaces, refer to Chapter 3 “Programmers Model” of the PrimeCell High-Performance Matrix (PL301) Technical Reference Manual at the following link:

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0422d/DDI0422D_hpm_pl301_r1p2_ts.pdf

Figure 40-2. Exanple operation of LRG arbitration scheme



41. DMA Controller

This chapter explains the function and operation of the DMA Controller.

41.1 Overview of the DMA Controller

The Direct Memory Access (DMA) Controller implements DMA with little CPU intervention. The DMA Controller performs complex data transfers through 'I' DMA channels. This section describes the features and the block diagram of the DMA Controller.

Features of the DMA Controller

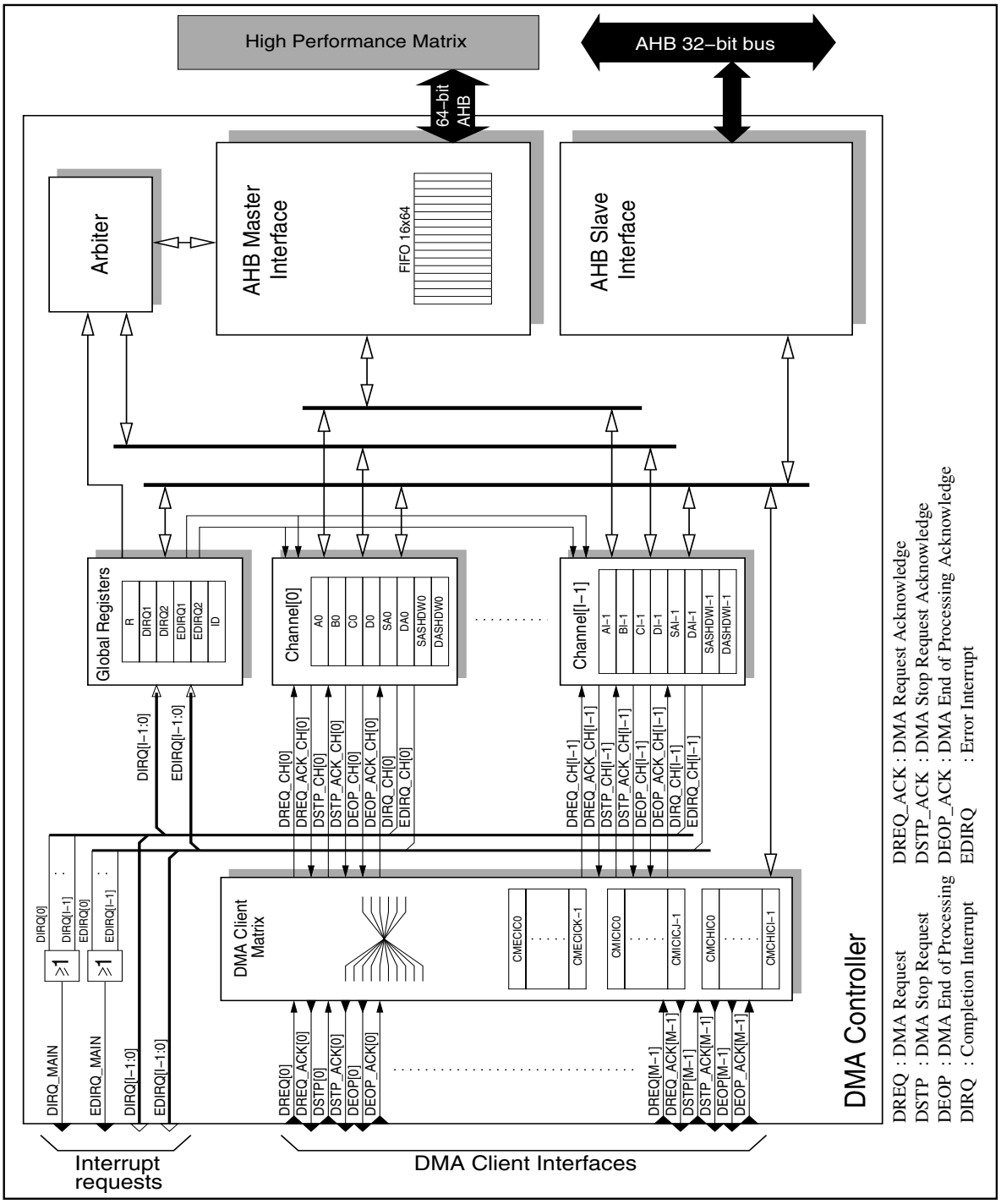
Features related to data transfer by the DMA Controller are:

- Data can be transferred independently over multiple channels
- A high number of MCU internal and MCU external DMA clients can be assigned to the available DMA channels
- Flexible priority between DMA channels (fixed, dynamic or round-robin)
- DMA transfer request sources
  - Hardware request (external and internal clients)
  - Software request (register write)
- Transfer modes
  - Block transfer, burst transfer and demand transfer
  - Addressing: full 32-bit (incrementing, decrementing or fixed)
  - Data types: 8-, 16-, 32- or 64-bit wide data

Block diagram of the DMA Controller

Figure 41-1 shows the top level block diagram of the DMA Controller.

Figure 41-1. Block diagram of DMA Controller



41.2 DMA Controller registers

The DMA Controller module has various registers to configure the DMA Controller global, DMA Controller channel and DMA Controller client matrix registers. All DMA Controller registers are explained in this section.

Registers of DMA Controller

The following registers are available for each instance of the DMA Controller:

- DMA Controller Global Configuration Register (DMAn_R)
- DMA Controller Global Completion Interrupt 1 Register (DMAn_DIRQ1)
- DMA Controller Global Completion Interrupt 2 Register (DMAn_DIRQ2)
- DMA Controller Global Error Interrupt 1 Register (DMAn_EDIRQ1)
- DMA Controller Global Error Interrupt 2 Register (DMAn_EDIRQ2)
- DMA Controller ID Register (DMAn_ID)
- DMA Controller Channel Configuration A Register Channel 'i' (DMAn_Ai)
- DMA Controller Channel Configuration B Register Channel 'i' (DMAn_Bi)
- DMA Controller Channel Configuration Source Address Register Channel 'i' (DMAn_SAi)
- DMA Controller Channel Configuration Destination Address Register Channel 'i' (DMAn_DAi)
- DMA Controller Channel Configuration C Register Channel 'i' (DMAn_Ci)
- DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di)
- DMA Controller Channel Configuration Source Address Shadow Register Channel 'i' (DMAn_SASHDWi)
- DMA Controller Channel Configuration Destination Address Shadow Register Channel 'i' (DMAn_DASHDWi)
- DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk)
- DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj)
- DMA Controller Client Matrix Channel Interface Configuration Register 'i' (DMAn_CMCHICi)

Notes:

- 'i' in the register name indicates that there is one instance of this register per DMA Controller instance
- 'n' in the register name indicates that there is one instance of this register per channel
- 'k' in the register name indicates that there is one instance of this register per MCU external client
- 'j' in the register name indicates that there is one instance of this register per MCU internal client

Memory layout of DMA Controller registers

Table 41-1. Memory layout of DMA Controller registers

Offset	+3	+2	+1	+0
base address + i*0x40 + 0x0	DMAn_Ai 00000000 00001111 00000000 00000000			
base address + i*0x40 + 0x4	DMAn_Bi 00000000 00000000 00110011 01111111			
base address + i*0x40 + 0x8	DMAn_SAi 00000000 00000000 00000000 00000000			
base address + i*0x40 + 0xC	DMAn_DAi 00000000 00000000 00000000 00000000			
base address + i*0x40 + 0x10	DMAn_Ci XXXXXXXX XXXXXXXX 00000000 00000000			
base address + i*0x40 + 0x14	DMAn_Di 00000000 XXXXXXXX 00000000 XXXXXXXX			
base address + i*0x40 + 0x18	DMAn_SASHDWi 00000000 00000000 00000000 00000000			
base address + i*0x40 + 0x1C	DMAn_DASHDWi 00000000 00000000 00000000 00000000			
base address + i*0x40 + 02x0 - base address + i*0x40 + 0x3C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
base address + 0x00001000	DMAn_R 01000000 XXXXXXXX XXXXXXXX 00000001			
base address + 0x00001004	DMAn_DIRQ1 00000000 00000000 00000000 00000000			
base address + 0x00001008	DMAn_DIRQ2 00000000 00000000 00000000 00000000			

Table 41-1. Memory layout of DMA Controller registers

Offset	+3	+2	+1	+0
base address + 0x0000100C	DMAn_EDIRQ1 00000000 00000000 00000000 00000000			
base address + 0x00001010	DMAn_EDIRQ2 00000000 00000000 00000000 00000000			
base address + 0x00001014	DMAn_ID 00000000 00000000 00000000 00000000			
base address + 0x00001018 - base address + 0x00001FFC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
base address + 0x2000 + k*0x4	DMAn_CMECICk 00000000 00000000 XXXXXXXX 00000000			
base address + 0x2020 + j*0x4	DMAn_CMICICj 00000000 XXXXXXXX XXXXXXXX XXXXXXXX			
base address + 0x2800 + i*0x4	DMAn_CMCHICi XXXXXXXX XXXXXXXX 00000000 0000iiii			
base address + 0x00002900 to base address + 0x00003FFC	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			

Notes:

- 'n' in the register name indicates DMA Controller instance 'n'. 'n' = 0 ... N-1. Refer to the datasheet for the value of N
- 'i' in the register name indicates that the register is an instance 'i' of this register in the DMA Controller, 'i' = 0 ... I-1. Refer to the datasheet for the value of I
- 'k' in the register name indicates that the register is an instance 'k' of this register in the DMA Controller, 'k' = 0 ... K-1. Refer to the datasheet for the value of K
- 'j' in the register name indicates that the register is an instance 'j' of this register in the DMA Controller, 'j' = 0 ... J-1. J = M - 8. Refer to the datasheet for the value of M

41.2.1 DMA Controller Global Configuration Register (DMAn_R)

This register handles the enabling and halting of the entire DMA Controller, arbitration scheme of the DMA channel arbiter can be chosen, enabling of debug function, and the DMACs behavior in case of a 'debug event' can be selected.

DMA Controller Global Configuration Register (DMAn_R)

Figure 41-2. DMA Controller Global Configuration Register (DMAn_R)

DMAn_R																															
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**
DE	DSHR	DBE	PR[1]	PR[0]	DH	DB[1]	DB[0]	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	DSHS
RWP	R	RWP	RWP	RWP	RWP	RWP	RWP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	R
0	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 41-2. DMA Controller Global Configuration Register (DMAn_R) bits

Bit Position	Bit Field Name	Bit Description
[31]	DE	DMA Enable If this bit is set to '0', the DMA Controller is disabled. This also means that all DMA channels are disabled independent of the settings of DMAn_Ai:EB. If this bit is set to '1', the DMA Controller is enabled and the enabling of the channels depend on the setting of DMAn_Ai:EB. When this bit is set to '0' during a DMA transfer, the channel which is in the middle of a transfer stops at the next transfer gap. The transfer gap means that DMA Controller deasserts the bus request to the bus arbiter for a short time in the middle of a DMA transfer after a block of data has been transferred. This ensures that the bus is not completely blocked by a very long DMA transfer. '0': DMA Controller globally disabled (initial value) '1': DMA Controller globally enabled

Table 41-2. DMA Controller Global Configuration Register (DMAn_R) bits

Bit Position	Bit Field Name	Bit Description
[30]	DSHR	DMA Stop/Halt Request Flag This bit indicates that the DMA transfers of all channels have been requested to halt or disable. This bit is set to '0' by hardware if none of the conditions below are true. This bit is set to '1' by hardware if one or more of the conditions below are true. Conditions for DMA Stop/Halt Request Flag: ■ DMAn_R:DE is set to '0' (all channels are disabled) ■ DMAn_R:DH is set to '1' (all channels are halted) ■ DMAn_R:DBE is set to '1', DMAn_R:DB is set to '10' (stop on debug events) and a debug event is pending. ■ DMAn_R:DBE is set to '1', DMAn_R:DB is set to '01' (halt on debug events) and a debug event is pending. '0': Indicates that global halt/disable condition of DMA Controller is removed '1': Indicates that DMA transfers of all channels are requested to halt or disable (initial value)
[29]	DBE	Debug Enable This bit determines whether DMA Controller reacts on a debug event (i.e. debugger break point). The behaviour of the DMA Controller on the occurence of a debug event depends on configuration bits, Debug Behaviour (DMAn_R:DB). '0': DMA Controller does not react on debug events (initial value) '1': DMA Controller reacts on debug events. Reaction depends on the setting of Debug Behaviour (DMAn_R:DB).
[28:27]	PR	Priority Type These bits select the arbitration scheme of the DMA Controller arbiter. In case of dynamic priority, channel priority is updated at the transfer gap. '00': Fixed priority (initial value) '01': Dynamic priority '10': Round robin '11': Reserved

Table 41-2. DMA Controller Global Configuration Register (DMAn_R) bits

Bit Position	Bit Field Name	Bit Description
[26]	DH	DMA Halt When this bit is set to a '1', all DMA channels are halted and do not perform DMA transfers until this bit is set back to '0'. After it is cleared the halted DMA transfers continue at the point they were halted. If this bit is set to '1' while a DMA transfer is ongoing, DMA Controller halts the transfer at the next transfer gap. Refer to the description of the DMAn_R:DE bit for more information about the transfer gap.
[25:24]	DB	Debug Behaviour '00': DMA Controller continues on a debug event (initial value) '01': DMA Controller halts all transfers on a debug event '10': DMA Controller stops all transfers on a debug event '11': Reserved
[23:8]	reserved	-
[7:1]	read0	-
[0]	DSHS	DMA Stop/Halt Status Flag '0': Indicates that a DMA transfer of at least one channel is running '1': Indicates that DMA transfers of all channels are halted or disabled (initial value)

41.2.2 DMA Controller Global Completion Interrupt 1 Register (DMAn_DIRQ1)

The DMA Controller Global Completion Interrupt 1 Register combines the completion Interrupt Flags (DMAn_Bi:DQ) from DMA channels 0 to 31.

DMA Controller Global Completion Interrupt 1 Register (DMAn_DIRQ1)

Figure 41-3. DMA Controller Global Completion Interrupt 1 Register (DMAn_DIRQ1)

DMAn_DIRQ1																															
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**
DIRQ[31]	DIRQ[30]	DIRQ[29]	DIRQ[28]	DIRQ[27]	DIRQ[26]	DIRQ[25]	DIRQ[24]	DIRQ[23]	DIRQ[22]	DIRQ[21]	DIRQ[20]	DIRQ[19]	DIRQ[18]	DIRQ[17]	DIRQ[16]	DIRQ[15]	DIRQ[14]	DIRQ[13]	DIRQ[12]	DIRQ[11]	DIRQ[10]	DIRQ[9]	DIRQ[8]	DIRQ[7]	DIRQ[6]	DIRQ[5]	DIRQ[4]	DIRQ[3]	DIRQ[2]	DIRQ[1]	DIRQ[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-3. DMA Controller Global Completion Interrupt 1 Register (DMAn_DIRQ1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DIRQ	Global Completion Interrupt 1 This is a read-only register which gives the DIRQ status of channels 0 to 31. Channels which are not available on a particular device read '0'.

41.2.3 DMA Controller Global Completion Interrupt 2 Register (DMAn_DIRQ2)

The DMA Controller Global Completion Interrupt 2 Register combines the Completion Interrupt Flags (DMAn_Bi:DQ) from DMA channels 32 to 63.

DMA Controller Global Completion Interrupt 2 Register (DMAn_DIRQ2)

Figure 41-4. DMA Controller Global Completion Interrupt 2 Register (DMAn_DIRQ2)

DMAn_DIRQ2																															
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**
DIRQ[31]	DIRQ[30]	DIRQ[29]	DIRQ[28]	DIRQ[27]	DIRQ[26]	DIRQ[25]	DIRQ[24]	DIRQ[23]	DIRQ[22]	DIRQ[21]	DIRQ[20]	DIRQ[19]	DIRQ[18]	DIRQ[17]	DIRQ[16]	DIRQ[15]	DIRQ[14]	DIRQ[13]	DIRQ[12]	DIRQ[11]	DIRQ[10]	DIRQ[9]	DIRQ[8]	DIRQ[7]	DIRQ[6]	DIRQ[5]	DIRQ[4]	DIRQ[3]	DIRQ[2]	DIRQ[1]	DIRQ[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-4. DMA Controller Global Completion Interrupt 2 Register (DMAn_DIRQ2) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DIRQ	Global Completion Interrupt 2 This is a read-only register which gives the DIRQ status of channels 32 to 63. Channels which are not available on a particular device read '0'.

41.2.4 DMA Controller Global Error Interrupt 1 Register (DMAn_EDIRQ1)

The DMA Controller Global Error Interrupt 1 Register combines the Error Interrupt Flags (DMAn_Bi:EQ) from DMA channels 0 to 31.

DMA Controller Global Error Interrupt 1 Register (DMAn_EDIRQ1)

Figure 41-5. DMA Controller Global Error Interrupt 1 Register (DMAn_EDIRQ1)

DMAn_EDIRQ1																															
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**
EDIRQ[31]	EDIRQ[30]	EDIRQ[29]	EDIRQ[28]	EDIRQ[27]	EDIRQ[26]	EDIRQ[25]	EDIRQ[24]	EDIRQ[23]	EDIRQ[22]	EDIRQ[21]	EDIRQ[20]	EDIRQ[19]	EDIRQ[18]	EDIRQ[17]	EDIRQ[16]	EDIRQ[15]	EDIRQ[14]	EDIRQ[13]	EDIRQ[12]	EDIRQ[11]	EDIRQ[10]	EDIRQ[9]	EDIRQ[8]	EDIRQ[7]	EDIRQ[6]	EDIRQ[5]	EDIRQ[4]	EDIRQ[3]	EDIRQ[2]	EDIRQ[1]	EDIRQ[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-5. DMA Controller Global Error Interrupt 1 Register (DMAn_EDIRQ1) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EDIRQ	Global Error Interrupt 1 This is a read-only register which gives the EDIRQ status of channels 0 to 31. Channels which are not available on a particular device read '0'.

41.2.5 DMA Controller Global Error Interrupt 2 Register (DMAn_EDIRQ2)

The DMA Controller Global Error Interrupt 2 Register combines the Error Interrupt Flags (DMAn_Bi:EQ) from DMA channels 32 to 63.

DMA Controller Global Error Interrupt 2 Register (DMAn_EDIRQ2)

Figure 41-6. DMA Controller Global Error Interrupt 2 Register (DMAn_EDIRQ2)

DMAn_EDIRQ2																															
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**
EDIRQ[31]	EDIRQ[30]	EDIRQ[29]	EDIRQ[28]	EDIRQ[27]	EDIRQ[26]	EDIRQ[25]	EDIRQ[24]	EDIRQ[23]	EDIRQ[22]	EDIRQ[21]	EDIRQ[20]	EDIRQ[19]	EDIRQ[18]	EDIRQ[17]	EDIRQ[16]	EDIRQ[15]	EDIRQ[14]	EDIRQ[13]	EDIRQ[12]	EDIRQ[11]	EDIRQ[10]	EDIRQ[9]	EDIRQ[8]	EDIRQ[7]	EDIRQ[6]	EDIRQ[5]	EDIRQ[4]	EDIRQ[3]	EDIRQ[2]	EDIRQ[1]	EDIRQ[0]
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-6. DMA Controller Global Error Interrupt 2 Register (DMAn_EDIRQ2) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	EDIRQ	Global Error Interrupt 2 This is a read-only register which gives the EDIRQ status of channels 32 to 63. Channels which are not available on a particular device read '0'.

41.2.6 DMA Controller ID Register (DMAn_ID)

The DMA Controller ID Register contains the identification number, revision number and patch level of the module.

DMA Controller ID Register (DMAn_ID)

Figure 41-7. DMA Controller ID Register (DMAn_ID)

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

Table 41-7. DMA Controller ID Register (DMAn_ID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module Number Module ID This read-only register gives the unique module identification number of the DMA Controller module. The unique module ID number identifies the version of the DMA Controller module used in the MCU. For the MID number of the DMA Controller refer to the device-specific datasheet.

41.2.7 DMA Controller Channel Configuration A Register Channel 'i' (DMAn_Ai)

The register specifies whether to enable or halt a DMA channel. It specifies the source of a DMA transfer request and DMA transfer parameters Block Count, Transfer Count, Beat Limit, Alternate and Timeout.

DMA Controller Channel Configuration A Register Channel 'i' (DMAn_Ai)

Figure 41-8. DMA Controller Channel Configuration A Register Channel 'n' (DMAn_Ai)

DMAn_Ai																															
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**
EB	PB	ST	IS[1]	IS[0]	AL	BL[1]	BL[0]	BC[3]	BC[2]	BC[1]	BC[0]	TO[3]	TO[2]	TO[1]	TO[0]	TC[15]	TC[14]	TC[13]	TC[12]	TC[11]	TC[10]	TC[9]	TC[8]	TC[7]	TC[6]	TC[5]	TC[4]	TC[3]	TC[2]	TC[1]	TC[0]
RWP	RWP	R0WP1	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
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	0	0	0	0	0

Table 41-8. DMA Controller Channel Configuration A Register Channel 'n' (DMAn_Ai) bits

Bit Position	Bit Field Name	Bit Description
[31]	EB	Enable Bit This bit is used to enable/disable a DMA channel. If this bit is set to '1', the channel is enabled and waits for a request to start a DMA transfer (Before that, the DMAn_R:DE bit needs to be set to '1'). If this bit is set to '0', the channel is disabled and does not perform a DMA transfer. When this bit is set to '0' during a running DMA transfer which will not complete at the next transfer gap, the DMA transfer is terminated. This is regarded as a forced stop and an error interrupt is generated. When this bit is set to '0' while the last block of a DMA transfer is running, the DMA transfer completes at the next transfer gap and a completion interrupt is generated. About the transfer gap, refer to the description of DMAn_R:DE bit. This bit is useful to re-configure each configuration register of the channel after a DMA transfer. '0': Channel is disabled (initial value) '1': Channel is enabled

Table 41-8. DMA Controller Channel Configuration A Register Channel 'n' (DMAn_Ai) bits

Bit Position	Bit Field Name	Bit Description
[30]	PB	Pause Bit This bit is used to halt the transfer of the DMA channel. If it is set to '1', this channel halts the transfer and does not perform a DMA transfer until this bit is cleared. When this bit is set to '1' while no transfer is ongoing DMA Controller enters the halt state immediately. When it was set to '1' during a running transfer, the halt state is entered at the next transfer gap. If the DMA transfer completes at the next transfer gap a completion interrupt is issued. When this bit is set to '0' the halt condition is cleared and DMA Controller waits for the next request to continue the DMA transfer. This bit is useful to halt a DMA transfer without re-configuration of each channel configuration register. '0': Channel is not halted (initial value) '1': Channel is halted
[29]	ST	Software Trigger This bit is used to generate a software request. This bit can only be set when DMAn_Bi:SR = '1' and the stop status is either initial or normal. When this bit is set to '1', a DMA transfer is requested because a software request has been received. The DMA Controller sets this bit to '0' if the Software Trigger has been recognized, an internal channel request for service is set, and Software Trigger Ready (DMAn_Bi:SR) is set to '0'. The software request is successful when DMAn_Bi:SR changes its status from '1' to '0'. ST can only be read as '0'. '0': No software request (initial value) '1': Software request

Table 41-8. DMA Controller Channel Configuration A Register Channel 'n' (DMAn_Ai) bits

Bit Position	Bit Field Name	Bit Description
[28:27]	IS	Input Select These bits are used to select the trigger source of a DMA transfer request. When the trigger source of a DMA transfer is a software request, the IS bits are set to '00'. When the trigger source of a DMA transfer is a hardware request, the IS bits are set to '01'. '00': Software request (initial value) '01': Hardware request '10': Reserved '11': Reserved **Note:** When the transfer mode is a block transfer or burst transfer, edge detection is used. When the transfer mode is demand transfer,level detection is used. For external DMA clients the polarity can be configured in the DMA Controller Client Matrix External Client Interface Configuration Registers (DMAn_CMECICk).
[26]	AL	Alternate This bit decides whether the data transfers should alternate between read and write or should be contiguous reads followed by contiguous writes. The alternation takes place after each incremental read burst and after each single data read. '0': Contiguous (initial value) '1': Alternate
[25:24]	BL	Beat Limit Beat Limit controls the maximum burst length the AHB master can make on the AHB bus for this DMA channel. '00': Single transfer (SINGLE) (initial value) '01': 4-beat incrementing burst (INCR4) '10': 8-beat incrementing burst (INCR8) '11': 16-beat incrementing burst (INCR16) These bits are used in combination with Block Count to decide which type of burst has to be transmitted over the AHB interface.
[23:20]	BC	Block Count These bits specify the total length of a single block in block/burst transfer mode. The maximum block count is 16. For example, if BC = 4, the number of data transfers is 5 (BC + 1). Alternate, Block Count and Beat Limit together decide the bursts generated by the AHB master. In demand transfer mode, Block Count has a different meaning, here it is used to determine the maximum number of data transfers that can be made in one arbitration phase.

Table 41-8. DMA Controller Channel Configuration A Register Channel 'n' (DMAn_Ai) bits

Bit Position	Bit Field Name	Bit Description
[19:16]	TO	Timeout Timeout is applicable only in demand transfer mode. When signal DMA Request (DREQ) is de-asserted the Timeout count starts down counting on every DMA Controller clock cycle. If Timeout reaches zero, the DMA Controller finishes the current arbitration phase after completing the current data transfer, which allows the channel arbiter to arbitrate the next requesting channel.
[15:0]	TC	Transfer Count These bits are used to specify the transfer count for the block/burst/demand transfer. The maximum transfer count is 65536. If TC is set to zero then one transfer is done and if TC is set to 65535 then 65536 transfers are done. In burst and block transfer mode the TC represents the number of blocks of data transfers that the channel has to make before DMA 'End of Processing' (DEOP) is generated. For e.g. if TC = 9 and DMAn_Ai:BC = 9, then the total number of transfers the DMA Controller does = (DMAn_Ai:BC + 1) x (TC + 1) = 10 x 10 = 100 transfers before DEOP is generated. In demand transfer mode, TC represents the maximum number of data transfers that the channel has to make. For example if TC = 99, then the channel can make 100 DMA transfers after which DEOP is asserted.

41.2.8 DMA Controller Channel Configuration B Register Channel 'i' (DMAn_Bi)

This register contains Error and Completion Interrupt Flags, and Interrupt Mask bits. Stop status of DMA operation is available in this register. Operation mode selection, transfer width, source, and destination protection information are located here. Software trigger ready flag shows readiness of DMA channel to receive a Software Trigger. Priority Number for DMA channel arbiter can be set here for fixed and dynamic priority arbitration scheme.

DMA Controller Channel Configuration B Register Channel 'i' (DMAn_Bi)

Figure 41-9. DMA Controller Channel Configuration B Register Channel 'i' (DMAn_Bi)

DMAn_Bi																															
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**
DQ	EQ	MS[1]	MS[0]	TW[1]	TW[0]	SR	read0	read0	read0	read0	EI	CI	SS[2]	SS[1]	SS[0]	SP[3]	SP[2]	SP[1]	SP[0]	DP[3]	DP[2]	DP[1]	DP[0]	read0	PN[6]	PN[5]	PN[4]	PN[3]	PN[2]	PN[1]	PN[0]
R	R	RWP	RWP	RWP	RWP	R	R0	R0	R0	R0	RWP	RWP	R	R	R	RWP	RWP	RWP	R1	RWP	RWP	RWP	R1	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1	1	1	1	1	1	1

Table 41-9. DMA Controller Channel Configuration B Register Channel 'i (DMAn_Bi) bits

Bit Position	Bit Field Name	Bit Description
[31]	DQ	Flag of DIRQ DQ is set to '1' when a DMA transfer has completed successfully. DQ is cleared by hardware if the DMAn_Ci:CD (Clear DIRQ) bit is set to '1'. Otherwise the value is retained.
[30]	EQ	Flag of EDIRQ EQ is set to '1' when a DMA transfer has finished with an error. EQ is cleared by hardware if the DMAn_Ci:CE (Clear EDIRQ) bit is set to '1'. Otherwise the value is retained.
[29:28]	MS	Mode Select MS sets the transfer mode of the channel. '00': Block transfer mode (initial value) '01': Burst transfer mode '10': Demand transfer mode '11': Reserved
[27:26]	TW	Transfer Width TW specifies the data width for every data transfer of the DMA transfer. '00': Byte (initial value) '01': Half word '10': Word '11': Double word

Table 41-9. DMA Controller Channel Configuration B Register Channel 'i (DMAn_Bi) bits

Bit Position	Bit Field Name	Bit Description
[25]	SR	Software Trigger Ready This bit is used to signal that the DMA channel is ready to receive a software request. A number of conditions (listed below) can be responsible if the DMA channel is not ready to receive a software request. If one or more of these conditions are true, then SR is set to '0' by hardware. If none of the conditions are true, then SR is set to '1' by hardware. ■ DMAn_R:DE == '0'; DMA Controller is disabled. ■ DMAn_R:DH == '1'; All DMA channels are halted. ■ DMAn_Ai:EB == '0'; DMA channel is disabled. ■ DMAn_Ai:PB == '1'; DMA channel is halted. ■ DMAn_Ai:IS != '00'; Input select is not set to software. ■ A Debug Event is pending, DMAn_R:DBE == '1' and (DMAn_R:DB == '01' or DMAn_R:DB == '10'). A Debug Event is pending while Debug Enable is set and Debug Behaviour is set to HALT or STOP. DMA transfer is active and a software request is pending.
[24:21]	read0	-
[20]	EI	Error Interrupt Enable This bit is used to control the issue of an error interrupt (EDIRQ). '0': Error interrupt issuance is disabled (initial value) '1': Error interrupt issuance is enabled If this bit is set to '1', an error interrupt is issued due to any of the following transfer errors: ■ Transfer stop request by the DSTP signal or disable the transfer with DMAn_Ai:EB or DMAn_R:DE, or debug event (if DMAn_R:DBE == '1' and DMAn_R:DB[1:0] == '10'). ■ Source access error. ■ Destination access error.
[19]	CI	Completion Interrupt Enable This bit is used to control the issue of a completion interrupt (DIRQ). If this bit is set to '1', a completion interrupt is issued after the DMA transfer is completed normally. '0': Completion interrupt issuance is disabled (initial value) '1': Completion interrupt issuance is enabled

Table 41-9. DMA Controller Channel Configuration B Register Channel 'i (DMAn_Bi) bits

Bit Position	Bit Field Name	Bit Description
[18:16]	SS	Stop Status These bits are used to show the end code of DMA transfer. SS is set by hardware when an error or completion interrupt is raised and it is cleared by hardware when either DMAn_Ci:CE or DMAn_Ci:CD is set to '1'. '000': Initial value (Status type - None) '001': Reserved (Status type - n/a) '010': Stop Request (Status type - Stop) by: ■ DSTP ■ Channel disable (DMAn_Ai:EB = '0') ■ DMA disable (DMAn_R:DE = '0') ■ Debug event (DMAn_R:DBE = '1' and DMAn_R:DB = '10') '011': Source access error (Status type - Error) '100': Destination access error (Status type - Error) '101': Normal end (Status type - End) '110': Reserved (Status type - n/a) '111': Reserved (Status type - n/a) When different events occur at the same time, the end code is displayed according to the following priority: Highest Priority: ■ Reset ■ Cleared by clearing completion/error interrupt (DIRQ/EDIRQ) ■ Source access error ■ Destination access error Lowest Priority: ■ Stop request ■ **Note:** Note: If the interrupt bit is cleared then Stop Status is also cleared.
[15:12]	SP	Source Protection These bits are used to control source access protection. During source accesses HPROTM is driven with SP during the AHB address phase. SP[3] - '0': Not cacheable (initial value) SP[3] - '1': Cacheable SP[2] - '0': Not bufferable (initial value) SP[2] - '1': Bufferable SP[1] - '0': User access SP[1] - '1': Privileged access (initial value) SP[0] - '0': Instruction access (DMA Controller only makes data accesses thus SP[0] is fixed to '1' and writing it to '0' has no effect.) SP[0] - '1': Data access (initial value) SP is considered by sources which support protection control function.

Table 41-9. DMA Controller Channel Configuration B Register Channel 'i (DMAn_Bi) bits

Bit Position	Bit Field Name	Bit Description
[11:8]	DP	Destination Protection These bits are used to control destination access protection. During destination accesses HPROTM is driven with DP during the AHB address phase. DP[3] - '0': Not cacheable (initial value) DP[3] - '1': Cacheable DP[2] - '0': Not bufferable (initial value) DP[2] - '1': Bufferable DP[1] - '0': User access DP[1] - '1': Privileged access (initial value) DP[0] - '0': Instruction access (DMA Controller only makes data accesses thus DP[0] is fixed to '1' and writing it to '0' has no effect.) DP[0] - '1': Data access (initial value) DP is only considered by destinations which support protection control function.
[7]	read0	-
[6:0]	PN	Priority Number These bits are used to specify the Priority Number (PN). The PN is needed if fixed priority or dynamic priority has been selected as the arbitration scheme. It has no significance if the round robin arbitration scheme is selected. The channel with lower priority number value has a higher priority.

41.2.9 DMA Controller Channel Configuration Source Address Register Channel 'i' (DMAn_SAi)

This register holds the source address for the DMA transfer of channel 'i'. There are 'I' identical registers for channel 0 to 'I-1'.

DMA Controller Channel Configuration Source Address Register Channel 'i' (DMAn_SAi)

Figure 41-10. DMA Controller Channel Configuration Source Address Register Channel 'i' (DMAn_SAi)

DMAn_SAi																															
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**
SA[31]	SA[30]	SA[29]	SA[28]	SA[27]	SA[26]	SA[25]	SA[24]	SA[23]	SA[22]	SA[21]	SA[20]	SA[19]	SA[18]	SA[17]	SA[16]	SA[15]	SA[14]	SA[13]	SA[12]	SA[11]	SA[10]	SA[9]	SA[8]	SA[7]	SA[6]	SA[5]	SA[4]	SA[3]	SA[2]	SA[1]	SA[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-10. DMA Controller Channel Configuration Source Address Register Channel 'i' (DMAn_SAi) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SA	Source Address These bits are used to specify the Source Address (SA) to start a DMA transfer. SA is updated with the value of DMAn_SASHDWi at the end of a DMA transfer if Update Source Address (DMAn_Di:US) is set to '1', Fixed Block Source Address (DMAn_Di:FBS) is set to '0', and the transfer ends successfully. Otherwise it retains its value. **Note:** Warning: The DMAn_SAi register must be loaded with aligned addresses with respect to the transfer width. If non- aligned addresses are loaded the DMA Controller converts it to an aligned address according to the setting of Transfer Width (DMAn_Bi:TW).

41.2.10 DMA Controller Channel Configuration Destination Address Register Channel 'i' (DMAn_DAi)

This register holds the destination address for the DMA transfer of Channel 'i'. There are 'I' identical registers for channel 0 to 'I-1'.

DMA Controller Channel Configuration Destination Address Register Channel 'i' (DMAn_DAi)

Figure 41-11. DMA Controller Channel Configuration Destination Address Register Channel 'i' (DMAn_DAi)

DMAn_DAi																															
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**
DA[31]	DA[30]	DA[29]	DA[28]	DA[27]	DA[26]	DA[25]	DA[24]	DA[23]	DA[22]	DA[21]	DA[20]	DA[19]	DA[18]	DA[17]	DA[16]	DA[15]	DA[14]	DA[13]	DA[12]	DA[11]	DA[10]	DA[9]	DA[8]	DA[7]	DA[6]	DA[5]	DA[4]	DA[3]	DA[2]	DA[1]	DA[0]
RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-11. DMA Controller Channel Configuration Destination Address Register Channel 'i' (DMAn_DAi)

Bit Position	Bit Field Name	Bit Description
[31:0]	DA	Destination Address These bits are used to specify the Destination Address (DA) to start a DMA transfer. DA is updated with the value of DMAn_DASHDWi at the end of a DMA transfer if Update Destination Address (DMAn_Di:UD) is set to '1', Fixed Block Destination Address (DMAn_Di:FBD) is set to '0', and the transfer ends successfully. Otherwise it retains its value. **Note:** Warning: The DMAn_DAi register must be loaded with aligned addresses with respect to the transfer width. If non- aligned addresses are loaded the DMA Controller converts it to an aligned address according to the setting of Transfer Width (DMAn_Bi:TW).

41.2.11 DMA Controller Channel Configuration C Register Channel 'i' (DMAn_Ci)

The DMA Controller Channel Configuration C Register is used to clear the Interrupt Flags set in the DMAi_Bn register. By writing a '1' to a bit in this register, the software can clear the corresponding flag in the DMAi_Bn register. There are 'I' identical registers for channel 0 to 'I-1'.

DMA Controller Channel Configuration C Register Channel 'i' (DMAn_Ci)

Figure 41-12. DMA Controller Channel Configuration C Register Channel 'i' (DMAn_Ci)

DMAn_Ci																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	CE	read0	read0	read0	read0	read0	read0	read0	CD
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	WP1	R0	R0	R0	R0	R0	R0	R0	WP1
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 41-12. DMA Controller Channel Configuration C Register Channel 'i' (DMAn_Ci) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	reserved	-
[15:9]	read0	-
[8]	CE	Clear EDIRQ Setting this bit clears the EDIRQ flag bit (DMAn_Bi:EQ). DMA Controller clears this bit automatically.
[7:1]	read0	-
[0]	CD	Clear DIRQ Setting this bit clears the DIRQ flag bit (DMAn_Bi:DQ). DMA Controller clears this bit automatically.

41.2.12 DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di)

This register controls the DMA channels addressing behaviour for a DMA transfer. Addressing can be independently chosen for the source and destination.

DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di)

Figure 41-13. DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di)

DMAn_Di																															
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**
FS	DES	US	FBS	read0	read0	read0	read0	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	FD	DED	UD	FBD	read0	read0	read0	read0	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
RWP	RWP	RWP	RWP	R0	R0	R0	R0	-	-	-	-	-	-	-	-	RWP	RWP	RWP	RWP	R0	R0	R0	R0	-	-	-	-	-	-	-	-
0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X

Table 41-13. DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di) bits

Bit Position	Bit Field Name	Bit Description
[31]	FS	Fixed Source Address This bit is used to keep the source address at a fixed value. '0': Source address is incremented (initial value) '1': Source address is kept fixed. The AHB master only makes single transfers to access the source
[30]	DES	Decrement Source Address If this bit is set the source address on the AHB interface is decremented on each AHB transfer. In this mode the AHB master makes only single transfers to access the source. '0': Source address is incremented (initial value) '1': Source address is decremented **Note:** Fixed Source Address (DMAn_Di:FS) has higher priority than Decrement Source Address. This bit is only effective if DMAn_Di:FS = '0' and DMAn_Di:FBS = '0'.
[29]	US	Update Source Address '0': DMAn_SAi is not updated after the DMA transfer is successfully completed. DMAn_SAi is retained (initial value) '1': DMAn_SAi is updated with the next address after the DMA transfer is successfully completed. E.g. last source address is 0x0BF000FC **Note:** This bit is only effective if DMAn_Di:FBS = '0'

Table 41-13. DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di) bits

Bit Position	Bit Field Name	Bit Description
[28]	FBS	Fixed Block Source Address '0': The Start Address of the first block of DMA transfer is set to the value stored in DMAn_SAi. The Start Address of consecutive blocks is the address following the previous block last address (according to Transfer Width (DMAn_Bi:TW)) (initial value) '1': The Start Address of each block is set to the value stored in DMAn_SAi **Note:** Setting of FBS is only effective if DMAn_Di:FS = '0' and (DMAn_Bi:MS = '00' (block transfer mode) or DMAn_Bi:MS = '01' (burst transfer mode)).
[27:24]	read0	-
[23:16]	reserved	-
[15]	FD	Fixed Destination Address This bit is used to keep the destination address at a fixed value. '0': Destination address is incremented (initial value) '1': Destination address is kept fixed. The AHB master only makes single transfers to access the destination
[14]	DED	Decrement Destination Address If this bit is set the destination address on the AHB interface is decremented on each AHB transfer. In this mode the AHB master makes only single transfers to access the destination. '0': Destination address is incremented (initial value) '1': Destination address is decremented **Note:** Fixed Destination Address (DMAn_Di:FD) has higher priority than Decrement Destination Address. This bit is only effective if DMAn_Di:FD = '0' and DMAn_Di:FBD = '0'.
[13]	UD	Update Destination Address '0': DMAn_DAi is not updated after the DMA transfer has success-fully completed. DMAn_DAi is retained (initial value) '1': DMAn_DAi is updated with the next address after the DMA transfer has successfully completed. E.g. last destination address was 0x0BF40010, DMAn_Bi:TW = 10 (word) then DMAn_DAi is updated with address 0x0BF40014. **Note:** This bit is effective only if DMAn_Di:FBD = '0'

Table 41-13. DMA Controller Channel Configuration D Register Channel 'i' (DMAn_Di) bits

Bit Position	Bit Field Name	Bit Description
[12]	FBD	Fixed Block Destination Address '0': The Start Address of first block of the DMA transfer is set to the value stored in DMAn_DAi. The Start Address of consecutive blocks is the address following the previous block's last address (according to Transfer Width (DMAn_Bi:TW) (initial value) '1': Start address of each block is set to the value stored in DMAn_DAi **Note:** Setting of FBD is only effective if FD = '0' and (DMAn_Bi:MS = '00' (block transfer mode) or DMAn_Bi:MS = '01' (burst transfer mode)).
[11:8]	read0	-
[7:0]	reserved	-

41.2.13 DMA Controller Channel Configuration Source Address Shadow Register Channel 'i' (DMAn_SASHDWi)

This register holds the address of the last source data transfer for the DMA transfer of channel '1' in case of a source access error. If the DMA transfer is stopped it contains the next address following the last successful data transfer. There are 'I' identical registers for channel 0 to 'I-1'.

DMA Controller Channel Configuration Source Address Shadow Register Channel 'i' (DMAn_SASHDWi)

Figure 41-14. DMA Controller Channel Configuration Source Address Shadow Register Channel 'i' (DMAn_SASHDWi)

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

Table 41-14. DMA Controller Channel Configuration Source Address Shadow Register Channel 'i' (DMAn_SASHDWi) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	SASHDW	Source Address Shadow At the start of a DMA transfer this register contains a copy of DMAn_SAi. During a DMA transfer it is either incremented, decremented, or stays constant according to the settings of DMAn_Di:FS, DMAn_Di:DES, DMAn_Di:US, and DMAn_Di:FBS. In case of a source access error SASHDW shows the address of the read access which causes the error. In case of a stop of the DMA transfer it shows the next address following the last read access done. **Note:** Warning: SASHDW does not show an aligned address if a non-aligned address is loaded into DMAn_SAi initially. However the DMA Controller will have made the read access to address SASHDW aligned according to the setting of Transfer Width (DMAn_Bi:TW).

41.2.14 DMA Controller Channel Configuration Destination Address Shadow Register Channel 'i' (DMAn_DASHDWi)

This register holds the address of the last destination data transfer for the DMA transfer of channel 'i' in case of a destination access error. If the DMA transfer is stopped it contains the next address following the last successful data transfer. There are 'I' identical registers for channel 0 to 'I-1'.

DMA Controller Channel Configuration Destination Address Shadow Register Channel 'i' (DMAn_DASHDWi)

Figure 41-15. DMA Controller Channel Configuration Destination Address Shadow Register Channel 'i' (DMAn_DASHDWi)

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

Table 41-15. DMA Controller Channel Configuration Destination Address Shadow Register Channel 'i' (DMAn_DASHDWi) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DASHDW	Destination Address Shadow At the start of a DMA transfer this register contains a copy of DMAn_DAi. During a DMA transfer it is either incremented, decremented, or stays constant according to the settings of DMAn_Di:FD, DMAn_Di:DED, DMAn_Di:UD, and DMAn_Di:FBD. In case of a destination access error DASHDW shows the address of the write access which causes the error. In case of a stop of the DMA transfer it shows the next address following the last write access done. **Note:** Warning: DASHDW does not show an aligned address if a non-aligned address is loaded into DMAn_DAi initially. However the DMA Controller will have made the write access to address DASHDW aligned according to the setting of Transfer Width (DMAi_Bn:TW).

41.2.15 DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk)

This register controls the external client interface 'k'. Enable bits set the availability of the different request/acknowledge signal pairs. Level config bits set the signal level of all external client interface signals. The Behavioural Configuration bits determine the behaviour of the acknowledge signals in case the DMA client is not selected.

DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk)

Figure 41-16. DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk)

DMAn_CMECICk																															
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**
read0	read0	read0	read0	BEHSTPACK	read0	BEHREQACK	read0	read0	read0	LVLEOPACK	LVLEOP	LVLSTPACK	LVLSTP	LVLREQACK	LVLREQ	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	ENEOP	ENSTP	ENCI
R0	R0	R0	R0	RWP	R0	RWP	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	RWP	RWP	RWP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Table 41-16. DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27]	BEHSTPACK	Behaviour Stop Acknowledge BEHSTPACK defines the behaviour of the External DMA Client Interface k output signal DSTP_ACK[k] if the client interface is not selected in any of the Channel Configuration Registers (DMAn_CMCHICn) or DMAn_CMECICk:ENSTP = '0' or DMAn_CMECICk:ENCI = '0'. '0': DSTP_ACK[k] outputs an inactive logic level. an inactive logic level depends on the setting of DMAn_CMECICk:LVLSTPACK (initial value) '1': DSTP[k] connected to DSTP_ACK[k]. If DMAn_CMECICk:LVL- STP = '0' and DMAn_CMECICk:LVLSTPACK = '0' or DMAn_CMECICk:LVLSTP = '1' and DMAn_CMECICk:LVLST- PACK = '1' then DSTP_ACK[k] = DSTP[k]. If DMAn_CMECICk:LVLSTP = '0' and DMAn_CMECICk:LVLSTPACK = '1' or DMAn_CMECICk:LVLSTP = '1' and DMAn_CMECICk:LVLSTPACK = '0' then DSTP_ACK[k] = !DSTP[k]
[26]	read0	-

Table 41-16. DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk) bits

Bit Position	Bit Field Name	Bit Description
[25]	BEHREQACK	Behaviour Request Acknowledge BEHREQACK sets the behaviour of the External DMA Client Interface k output signal DREQ_ACK[k] if the client interface is not selected in any of the Channel Configuration Registers (DMAn_CMCHICn) or DMAn_CMECICk:ENCI = '0'. '0': DREQ_ACK[k] outputs an inactive logic level. An inactive logic level depends on the setting of DMAn_CMECICk:LVLREQACK (initial value) '1': DREQ[k] connected to DREQ_ACK[k]. If DMAn_CMECICk:LVL- REQ = '0' and DMAn_CMECICk:LVLREQACK = '0' or DMAn_CMECICk:LVLREQ = '1' and DMAn_CMECICk:LVL- REQACK = '1' then DREQ_ACK[k] = DREQ[k]. If DMAn_CMECICk:LVLREQ = '0' and DMAn_CMECICk:LVLREQACK = '1' or DMAn_CMECICk:LVLREQ = '1' and DMAn_CMECICk:LVLREQACK = '0' then DREQ_ACK[k] = !DREQ[k]
[24:22]	read0	-
[21]	LVLEOPACK	Level End of Processing Acknowledge LVLEOPACK sets the signal level of the External DMA Client Interface k input signal DEOP_ACK[k]. '0': High active (initial value) '1': Low active
[20]	LVLEOP	Level End of Processing LVLEOP sets the signal level of the External DMA client interface k output signal DEOP[k]. '0': High active (initial value) '1': Low active
[19]	LVLSTPACK	Level Stop Acknowledge LVLSTPACK sets the signal level of the External DMA Client Interface k output signal DSTP_ACK[k]. '0': High active (initial value) '1': Low active
[18]	LVLSTP	Level Stop Request LVLSTP sets the signal level of the External DMA Client Interface k input signal DSTP[k]. '0': High active (initial value) '1': Low active

Table 41-16. DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk) bits

Bit Position	Bit Field Name	Bit Description
[17]	LVLREQACK	Level Request Acknowledge LVLREQACK sets the signal level of the External DMA Client Interface k output signal DREQ_ACK[k]. '0': High active (initial value) '1': Low active
[16]	LVLREQ	Level Request LVLREQ sets the signal level of the External DMA Client Interface k input signal DREQ[k]. '0': High active (initial value) '1': Low active
[15:8]	reserved	-
[7:3]	read0	-
[2]	ENEOP	Enable End of Processing ENEOP enables the use of the External DMA Client Interface k signal pair DEOP[k], DEOP_ACK[k]. **Note:** The setting of ENEOP is only effective if DMAn_CMECICk:ENCI is set to '1'. If DMAn_CMECICk:ENCI is set to '0' the use of DEOP/DEOP_ACK is disabled. '0': Disabled. DEOP[k] is set to an inactive level (depends on the setting of DMAn_CMECICk:LVLEOP) (initial value) '1': Enabled
[1]	ENSTP	Enable Stop ENSTP enables the use of the External DMA Client Interface k signal pair DSTP[k], DSTP_ACK[k]. **Note:** The setting of ENSTP is only effective if DMAn_CMECICk:ENCI is set to '1'. If DMAn_CMECICk:ENCI is set to '0' the use of DSTP/DSTP_ACK is disabled. '0': Disabled. DSTP_ACK[k] is set to an inactive level (depends on the setting of DMAn_CMECICk:LVLSTPACK) or connected to DSTP[k] (depends on the settings of DMAn_CMECICk:BEHST- PACK, DMAn_CMECICk:LVLSTP, and DMAn_CMECICk:LVL-STPACK). Stop request signal towards the channel is set to an inactive level (initial value) '1': Enabled

Table 41-16. DMA Controller Client Matrix External Client Interface Configuration Register 'k' (DMAn_CMECICk) bits

Bit Position	Bit Field Name	Bit Description
[0]	ENCI	Enable Client Interface ENCI enables the External DMA Client Interface k. The disabled mode is used during the configuration of the DMA client matrix for the external DMA clients. The setting of ENCI enables the use of the signal pair DREQ/DREQ_ACK and the use of DMAn_CMECICk:ENSTP or DMAn_CMECICk:ENEOP settings. If ENCI is set to '1' and both DMAn_CMECICk:ENSTP and DMAn_CMECICk:ENEOP are set to '0', then only the signal pair DREQ/DREQ_ACK of this External DMA client interface is available. '0': Disabled. DREQ_ACK[k] is set to inactive level (depends on the setting of DMAn_CMECICk:LVLREQACK) or connected to DREQ[k] (depends on the settings of DMAn_CMECICk:BEHREQACK, DMAn_CMECICk:LVLREQ, and DMAn_CMECICk:LVLREQACK). DMA request signal towards channel is set to inactive level (initial value) '1': Enabled

41.2.16 DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj)

This register controls internal client interface 'j'. Behavioural configuration bits determine the behaviour of the acknowledge signals in case the DMA client is not selected.

DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj)

Figure 41-17. DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj)

DMAn_CMICICj																															
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**
read0	read0	read0	read0	BEHSTPACK	read0	BEHREQACK	read0	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
R0	R0	R0	R0	RWP	R0	RWP	R0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 41-17. DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27]	BEHSTPACK	Behaviour Stop Acknowledge BEHSTPACK sets the behaviour of the Internal DMA Client Interface j output signal DSTP_ACK[j] if the client interface is not selected in any of the Channel Configuration Registers (DMAn_CMCHICn). **Note:** This configuration bit is read0 if the associated peripheral and/or the stop functionality is not available in a particular device. For its availability refer to the device-specific datasheet. '0': DSTP_ACK[j] outputs an inactive logic level (initial value) '1': DSTP[j] connects directly to DSTP_ACK[j]
[26]	read0	-

Table 41-17. DMA Controller Client Matrix Internal Client Interface Configuration Register 'j' (DMAn_CMICICj) bits

Bit Position	Bit Field Name	Bit Description
[25]	BEHREQACK	Behaviour Request Acknowledge BEHREQACK sets the behaviour of the Internal DMA Client Interface j output signal DREQ_ACK[j] if the client interface is not selected in any of the Channel Configuration Registers (DMAn_CMCHICn). **Note:** This configuration bit is read0 if the associated peripheral is not available in a particular device. For its availability refer to the device-specific datasheet. '0': DREQ_ACK[j] outputs an inactive logic level (initial value) '1': DREQ[j] connects directly to DREQ_ACK[j]
[24]	read0	-
[23:0]	reserved	-

41.2.17 DMA Controller Client Matrix Channel Interface Configuration Register 'i' (DMAn_CMCHICi)

The DMA client routed to DMA channel 'i' is specified in this register.

DMA Controller Client Matrix Channel Interface Configuration Register 'i' (DMAn_CMCHICi)

Figure 41-18. DMA Controller Client Matrix Channel Interface Configuration Register 'i' (DMAn_CMCHICi)

DMAn_CMCHICi																															
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**
reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	read0	read0	read0	read0	read0	read0	read0	CI[8]	CI[7]	CI[6]	CI[5]	CI[4]	CI[3]	CI[2]	CI[1]	CI[0]
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R0	R0	R0	R0	R0	R0	R0	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP	RWP
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	i	i	i	i

Table 41-18. DMA Controller Client Matrix Channel Interface Configuration Register 'i' (DMAn_CMCHICi) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	reserved	-
[15:9]	read0	-
[8:0]	CI	Client Interface The Client Interface specifies that DMA channel ‘i’ is routed to the DMA client given by the binary value of CI. The initial value of CI is equal to the channel number ‘i’.

41.3 Operation of the DMA Controller

The following section describes the operation of the DMA Controller.

Features of the DMA Controller

- DMA Client Matrix, routing 'M' DMA clients to 'N' DMA channels
- DMA trigger
  - Hardware request (external and internal clients)
  - Software request
- Three transfer modes
  - Block transfer
  - Burst transfer
  - Demand transfer
- 8-, 16-, 32- or 64-bit wide data transfers
- Master transactions can be done in user or privileged mode
- Writing the configuration registers can only be done in privileged mode and 8-, 16- or 32-bit wide. Illegal accesses result in an error response
- Reading the configuration registers can be done in user or privileged mode
- Incrementing, decrementing or fixed addressing, independent of the source and destination
- Independent source and destination access protection
- The central interrupt flag register for completion and error interrupts
- Stop status per channel for analysis by ISRs or debugging
- Shadow registers for source and destination address
- Configurable debug event behaviour (continue, halt or stop)
- Three channel arbitration schemes
  - Fixed priority
  - Dynamic priority
  - Round-robin

Global functions of the DMA Controller

DMAC enabling or halting

After a reset the DMA Controller is disabled. The DMA Controller is enabled by setting DMA Enable (DMAn_R:DE) to '1'. If DMA Enable is set to '1' the individual DMA Channel Enable (DMAn_An:EB) settings become effective. Setting DMA Enable to '0' disables the complete DMA Controller. Setting DMA Enable to '0' with an uncompleted transfer running on a DMA channel, the running transfer is stopped when its current block of data has been transferred and an error interrupt is issued. In the event it is the last block of data of this DMA transfer, a completion interrupt is issued. DMA channels which have a DMA transfer pending, but are not currently served are disabled and an error interrupt is issued. DMA channels which are enabled but have not started a DMA transfer (because they are waiting for a transfer request) are simply disabled without issuing an error interrupt.

The DMA Controller can be halted completely (all DMA channels) by writing a single bit, DMA Halt (DMAn_R:DH). When this bit is set to '1', all DMA channels are requested to halt and not perform DMA transfers until this bit is cleared. After DMA Halt is cleared the halted DMA transfers continue from the point they were halted. If DMA Halt is set to '1' while a transfer is running, halting takes place once the current block of data of the running transfer has been transferred. Depending on the clock ratio of the DMAC/source clock domain and the DMA Controller/destination clock domain a considerable time can elapse between the time when the halt request was set and when the DMA Controller is actually halted.

Global disable and halt request condition of the DMA Controller is indicated by a DMA Stop/Halt Request Flag (DMAn_R:DSHR). DMA Stop/Halt Request Flag is '0' if none of the below disable or halt request conditions is true:

- DMAn_R:DE bit is set to '0'
- DMAn_R:DH bit is set to '1'

- DMAn_R:DBE = '1', DMAn_R:DB[1:0] = '10' (stop on debug event), and a 'debug event' is pending
- DMAn_R:DBE = '1', DMAn_R:DB[1:0] = '01' (halt on debug event), and a 'debug event' is pending

If any of the above conditions is true, the DMA Stop/Halt Request Flag is '1' indicating that DMA transfers of all channels are requested to halt or stop.

The condition that all channels are halted or have come to a stop after a global halt or stop request is indicated by the DMA Stop/Halt Status Flag (DMAn_R:DSHS). The DMA Stop/Halt Status Flag is '0' if one or more channels are not yet stopped or halted and '1' if all channels are stopped or halted.

DMA Controller debug behaviour

The DMA Controller can be configured to react in a predefined way on a 'debug event' (e.g. debugger break point). The feature to react on a 'debug event' can be enabled with the Debug Enable (DMAn_R:DBE) bit. If enabled the DMA Controllers behaviour depends on the setting of Debug Behaviour (DMAn_R:DB[1:0]). This feature is disabled after reset.

The behaviour can be configured to stop all transfers (DMAn_R:DB[1:0] = '10'), or to halt all transfers (DMAn_R:DB[1:0] = '01') or just to continue operation independent of debug events (DMAn_R:DB[1:0] = '00'). The Initial value is to continue operation independent of debug events.

41.3.1 DMA channels

41.3.1.1 Modes of operation

The DMA channels can operate in three modes:

- Block transfer mode
- Burst transfer mode
- Demand transfer mode

The mode must be set with the Mode Select (DMAn_Bi:MS[1:0]) bits. After reset, the channel is set to block transfer mode.

41.3.1.2 Block transfer mode

In block transfer mode the DMA client will request the transfers of a specified number of blocks of data. The number of blocks to be transferred is specified with Transfer Count (DMAn_Ai:TC[15:0]). Each block of data is transferred in one arbitration phase of the DMA arbiter. For each block to transfer a DMA transfer request, either a hardware or software request is needed. The DMA client or the software continues to give requests until the DMA Controller has transferred the specified number of blocks and thus the DMA transfer is completed. The DMA transfer will be successfully completed if all blocks of data were transferred without error or unsuccessfully completed if an error condition occurred.

After each transferred block of data the DMA arbiter does another arbitration and proceeds with the next requesting channel with the highest priority. The arbitration depends on the selected arbitration scheme and the set priorities of the requesting channels (for details about arbitration see 41.3.3 DMA arbiter).

DMA transfer requests

1. Hardware request

   For a channel hardware request, the Input Select bits (DMAn_Ai:IS) need to be set to '01'. The DMA client which is routed through the DMA Client Matrix to the channel will give the trigger by asserting DREQ. Refer to 41.3.2 DMA Client Matrix for the function and configuration of the DMA Client Matrix and to 41.3.5 External DMA Client Interface signal requirements for the DREQ requirements for an external DMA client.

2. Software request

   For a software request, the Input Select bits must to be set to '00' and the Software Trigger (DMAn_Ai:ST) must be set to '1'. The Software Trigger shall be set to '1' if the channel is ready to receive a software trigger, which is indicated with Software Trigger Ready (DMAn_Bi:SR) and no error condition is pending (DMAn_Bi:SS[2:0] is '000' or '101'). If the Software Trigger tries to set to '1' while Software Trigger Ready is '0', it is ignored. The Software Trigger is automatically cleared by hardware once the trigger is accepted or an error condition occurs. The error conditions can be:

   - The DMA channel is disabled right after setting Software Trigger
   - A debug event occurs and the DMA Controller is configured to stop all transfers on a debug event
   - The master interface receives an error response and the CPU sets the Software Trigger before receiving an error interrupt caused by the AHB error

Block of data

A block of data is determined by the setting of Block Count (DMAn_Ai:BC[3:0]) and Transfer Width (DMAn_Bi:TW[1:0]). The DMA Controller will make DMAn_Ai:BC + 1 data transfers from the source address range starting at Source Address (DMAn_SAi:SA) to destination address range starting at Destination Address (DMAn_DAi:DA). If DMAn_Ai:BC is set to '0' a single data transfer from DMAn_SAi:SA to DMAn_DAi:DA will be done. The settings for DMAn_Ai:BC[3:0], Beat Limit (DMAn_Ai:BL[1:0], Alternate (DMAn_Ai:AL) and DMAn_Bi:TW[1:0] define how the AHB master interface issues the DMAn_Ai:BC + 1 data transfers. The following table shows all possible combinations between DMAn_Ai:BC, DMAn_Ai:BL, and DMAn_Ai:AL. Only these three influence the sequence of AHB transfers, whereas DMAn_Bi:TW only affects the data size which will be transported.

Table 41-19. Block Count/Beat Limit/Alternate combinations

Block Count	Beat Limit	Alternate	Resulting sequence of AHB transfers
0	SINGLE	0	1x SINGLE RD + 1x SINGLE WR
1	SINGLE	0	2x SINGLE RD + 2x SINGLE WR
2	SINGLE	0	3x SINGLE RD + 3x SINGLE WR
3	SINGLE	0	4x SINGLE RD + 4x SINGLE WR
4	SINGLE	0	5x SINGLE RD + 5x SINGLE WR
5	SINGLE	0	6x SINGLE RD + 6x SINGLE WR
6	SINGLE	0	7x SINGLE RD + 7x SINGLE WR
7	SINGLE	0	8x SINGLE RD + 8x SINGLE WR
8	SINGLE	0	9x SINGLE RD + 9x SINGLE WR
9	SINGLE	0	10x SINGLE RD + 10x SINGLE WR
10	SINGLE	0	11x SINGLE RD + 11x SINGLE WR
11	SINGLE	0	12x SINGLE RD + 12x SINGLE WR
12	SINGLE	0	13x SINGLE RD + 13x SINGLE WR
13	SINGLE	0	14x SINGLE RD + 14x SINGLE WR
14	SINGLE	0	15x SINGLE RD + 15x SINGLE WR
15	SINGLE	0	16x SINGLE RD + 16x SINGLE WR
0	SINGLE	1	1x (1x SINGLE RD + 1x SINGLE WR)
1	SINGLE	1	2x (1x SINGLE RD + 1x SINGLE WR)
2	SINGLE	1	3x (1x SINGLE RD + 1x SINGLE WR)
3	SINGLE	1	4x (1x SINGLE RD + 1x SINGLE WR)
4	SINGLE	1	5x (1x SINGLE RD + 1x SINGLE WR)
5	SINGLE	1	6x (1x SINGLE RD + 1x SINGLE WR)
6	SINGLE	1	7x (1x SINGLE RD + 1x SINGLE WR)
7	SINGLE	1	8x (1x SINGLE RD + 1x SINGLE WR)
8	SINGLE	1	9x (1x SINGLE RD + 1x SINGLE WR)
9	SINGLE	1	10x (1x SINGLE RD + 1x SINGLE WR)
10	SINGLE	1	11x (1x SINGLE RD + 1x SINGLE WR)
11	SINGLE	1	12x (1x SINGLE RD + 1x SINGLE WR)
12	SINGLE	1	13x (1x SINGLE RD + 1x SINGLE WR)
13	SINGLE	1	14x (1x SINGLE RD + 1x SINGLE WR)
14	SINGLE	1	15x (1x SINGLE RD + 1x SINGLE WR)
15	SINGLE	1	16x (1x SINGLE RD + 1x SINGLE WR)
0	INCR4	0	1x SINGLE RD + 1x SINGLE WR
1	INCR4	0	2x SINGLE RD + 2x SINGLE WR
2	INCR4	0	3x SINGLE RD + 3x SINGLE WR
3	INCR4	0	1x 4_BEAT RD + 1x 4_BEAT WR
4	INCR4	0	1x 4_BEAT RD + 1x SINGLE RD + 1x 4_BEAT WR + 1x SINGLE WR
5	INCR4	0	1x 4_BEAT RD + 2x SINGLE RD + 1x 4_BEAT WR + 2x SINGLE WR

Table 41-19. Block Count/Beat Limit/Alternate combinations

Block Count	Beat Limit	Alternate	Resulting sequence of AHB transfers
6	INCR4	0	1x 4_BEAT RD + 3x SINGLE RD + 1x 4_BEAT WR + 3x SINGLE WR
7	INCR4	0	2x 4_BEAT RD + 2x 4_BEAT WR
8	INCR4	0	2x 4_BEAT RD + 1x SINGLE RD + 2x 4_BEAT WR + 1x SINGLE WR
9	INCR4	0	2x 4_BEAT RD + 2x SINGLE RD + 2x 4_BEAT WR + 2x SINGLE WR
10	INCR4	0	2x 4_BEAT RD + 3x SINGLE RD + 2x 4_BEAT WR + 3x SINGLE WR
11	INCR4	0	3x 4_BEAT RD + 3x 4_BEAT WR
12	INCR4	0	3x 4_BEAT RD + 1x SINGLE RD + 3x 4_BEAT WR + 1x SINGLE WR
13	INCR4	0	3x 4_BEAT RD + 2x SINGLE RD + 3x 4_BEAT WR + 2x SINGLE WR
14	INCR4	0	3x 4_BEAT RD + 3x SINGLE RD + 3x 4_BEAT WR + 3x SINGLE WR
15	INCR4	0	4x 4_BEAT RD + 4x 4_BEAT WR
0	INCR4	1	1x SINGLE RD + 1x SINGLE WR
1	INCR4	1	2x (1x SINGLE RD + 1x SINGLE WR)
2	INCR4	1	3x (1x SINGLE RD + 1x SINGLE WR)
3	INCR4	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR)
4	INCR4	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
5	INCR4	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
6	INCR4	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
7	INCR4	1	2x (1x 4_BEAT RD + 1x 4_BEAT WR)
8	INCR4	1	2x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
9	INCR4	1	2x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
10	INCR4	1	2x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
11	INCR4	1	3x (1x 4_BEAT RD + 1x 4_BEAT WR)
12	INCR4	1	3x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
13	INCR4	1	3x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
14	INCR4	1	3x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
15	INCR4	1	4x (1x 4_BEAT RD + 1x 4_BEAT WR)
0	INCR8	0	1x SINGLE RD + 1x SINGLE WR
1	INCR8	0	2x SINGLE RD + 2x SINGLE WR
2	INCR8	0	3x SINGLE RD + 3x SINGLE WR
3	INCR8	0	1x 4_BEAT RD + 1x 4_BEAT WR
4	INCR8	0	1x 4_BEAT RD + 1x SINGLE RD + 1x 4_BEAT WR + 1x SINGLE WR
5	INCR8	0	1x 4_BEAT RD + 2x SINGLE RD + 1x 4_BEAT WR + 2x SINGLE WR
6	INCR8	0	1x 4_BEAT RD + 3x SINGLE RD + 1x 4_BEAT WR + 3x SINGLE WR
7	INCR8	0	1x 8_BEAT RD + 1x 8_BEAT WR
8	INCR8	0	1x 8_BEAT RD + 1x SINGLE RD + 1x 8_BEAT WR + 1x SINGLE WR
9	INCR8	0	1x 8_BEAT RD + 2x SINGLE RD + 1x 8_BEAT WR + 2x SINGLE WR
10	INCR8	0	1x 8_BEAT RD + 3x SINGLE RD + 1x 8_BEAT WR + 3x SINGLE WR
11	INCR8	0	1x 8_BEAT RD + 1x 4_BEAT RD + 1x 8_BEAT WR + 1x 4_BEAT WR
12	INCR8	0	1x 8_BEAT RD + 1x 4_BEAT RD + 1x SINGLE RD + 1x 8_BEAT WR + 1x 4_BEAT WR + 1x SINGLE WR

Table 41-19. Block Count/Beat Limit/Alternate combinations

Block Count	Beat Limit	Alternate	Resulting sequence of AHB transfers
13	INCR8	0	1x 8_BEAT RD + 1x 4_BEAT RD + 2x SINGLE RD + 1x 8_BEAT WR + 1x 4_BEAT WR + 2x SINGLE WR
14	INCR8	0	1x 8_BEAT RD + 1x 4_BEAT RD + 3x SINGLE RD + 1x 8_BEAT WR + 1x 4_BEAT WR + 3x SINGLE WR
15	INCR8	0	2x 8_BEAT RD + 2x 8_BEAT WR
0	INCR8	1	1x SINGLE RD + 1x SINGLE WR
1	INCR8	1	2x (1x SINGLE RD + 1x SINGLE WR)
2	INCR8	1	3x (1x SINGLE RD + 1x SINGLE WR)
3	INCR8	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR)
4	INCR8	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
5	INCR8	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
6	INCR8	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
7	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR)
8	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
9	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
10	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
11	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR)
12	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
13	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
14	INCR8	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
15	INCR8	1	2x (1x 8_BEAT RD + 1x 8_BEAT WR)
0	INCR16	0	1x SINGLE RD + 1x SINGLE WR
1	INCR16	0	2x SINGLE RD + 2x SINGLE WR
2	INCR16	0	3x SINGLE RD + 3x SINGLE WR
3	INCR16	0	1x 4_BEAT RD + 1x 4_BEAT WR
4	INCR16	0	1x 4_BEAT RD + 1x SINGLE RD + 1x 4_BEAT WR + 1x SINGLE WR
5	INCR16	0	1x 4_BEAT RD + 2x SINGLE RD + 1x 4_BEAT WR + 2x SINGLE WR
6	INCR16	0	1x 4_BEAT RD + 3x SINGLE RD + 1x 4_BEAT WR + 3x SINGLE WR
7	INCR16	0	1x 8_BEAT RD + 1x 8_BEAT WR
8	INCR16	0	1x 8_BEAT RD + 1x SINGLE RD + 1x 8_BEAT WR + 1x SINGLE WR
9	INCR16	0	1x 8_BEAT RD + 2x SINGLE RD + 1x 8_BEAT WR + 2x SINGLE WR
10	INCR16	0	1x 8_BEAT RD + 3x SINGLE RD + 1x 8_BEAT WR + 3x SINGLE WR
11	INCR16	0	1x 8_BEAT RD + 1x 4_BEAT RD + 1x 8_BEAT WR + 1x 4_BEAT WR
12	INCR16	0	1x 8_BEAT RD + 1x 4_BEAT RD + 1x SINGLE RD + 1x 8_BEAT WR + 1x 4_BEAT WR + 1x SINGLE WR

Table 41-19. Block Count/Beat Limit/Alternate combinations

Block Count	Beat Limit	Alternate	Resulting sequence of AHB transfers
13	INCR16	0	1x 8_BEAT RD + 1x 4_BEAT RD + 2x SINGLE RD + 1x 8_BEAT WR + 1x 4_BEAT WR + 2x SINGLE WR
14	INCR16	0	1x 8_BEAT RD + 1x 4_BEAT RD + 3x SINGLE RD + 1x 8_BEAT WR + 1x 4_BEAT WR + 3x SINGLE WR
15	INCR16	0	1x 16_BEAT RD + 1x 16_BEAT WR
0	INCR16	1	1x SINGLE RD + 1x SINGLE WR
1	INCR16	1	2x (1x SINGLE RD + 1x SINGLE WR)
2	INCR16	1	3x (1x SINGLE RD + 1x SINGLE WR)
3	INCR16	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR)
4	INCR16	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
5	INCR16	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
6	INCR16	1	1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
7	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR)
8	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
9	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
10	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
11	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR)
12	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 1x SINGLE RD + 1x SINGLE WR
13	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 2x (1x SINGLE RD + 1x SINGLE WR)
14	INCR16	1	1x (1x 8_BEAT RD + 1x 8_BEAT WR) + 1x (1x 4_BEAT RD + 1x 4_BEAT WR) + 3x (1x SINGLE RD + 1x SINGLE WR)
15	INCR16	1	1x 16_BEAT RD + 1x 16_BEAT WR

Note:

i_BEAT RD can be an 'i' beat incremental burst (INCRi) or it can be 'i' times a single (SINGLE) data transfer. i_BEAT RD will be 'i' times a SINGLE transfer if one or more of the following conditions is met:

- Fixed Source Address (DMAn_Di:FS) is set to '1'
- Decrement Source Address (DMAn_Di:DES) is set to '1'
- The 1 KB AHB address boundary will be crossed by the read burst transaction

i_BEAT WR can be an 'i' beat incremental burst (INCRi) or it can be 'i' times a single (SINGLE) data transfer. i_BEAT WR will be 'n' times a SINGLE transfer if one or more of the following conditions is met:

- Fixed Destination Address (DMAn_Di:FD) is set to '1'
- Decrement Destination Address (DMAn_Di:DED) is set to '1'
- The 1 KB AHB address boundary will be crossed by the write burst transaction

After each successful read data transfer, the Source Address Shadow Register (DMAn_SASHDWi:SASHDW) will be either incremented, decremented or unaltered. The behaviour is determined by the settings of the Fixed Source Address, Decrement Source Address or Fixed Block Source Address. The Destination Address Shadow Register (DMAn_DASHDWi:DASHDW) exhibits the same behaviour with respect to the settings of the Fixed Destination Address, Decrement

Destination Address or Fixed Block Destination Address and will be updated after each successful write data transfer. The tables below list the possible combinations and resulting action.

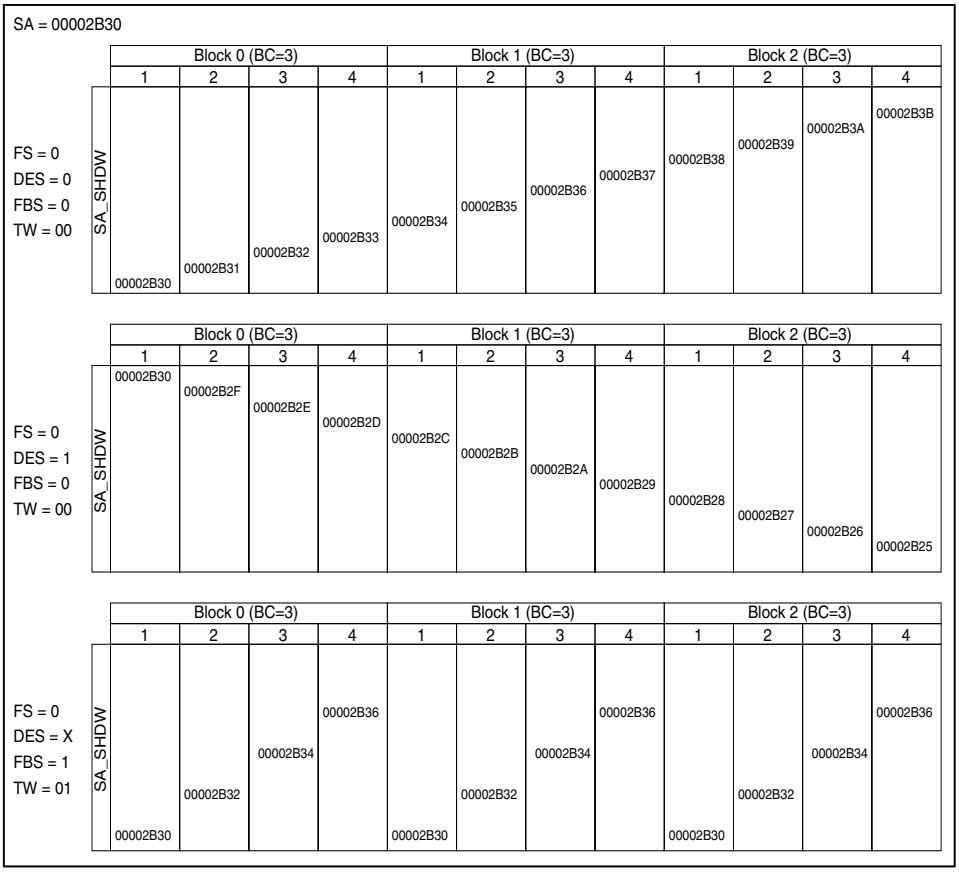
Table 41-20. Source Address Shadow update behaviour

DMAn_Di:FS	DMAn_Di:DES	DMAn_Di:FBS	Description of DMAn_SASHDWi:SASHDW update behaviour
'0'	'0'	'0'	SASHDW is incremented at each successful read data transfer. The size of the address increment depends on the Transfer Width.
'0'	'1'	'0'	SASHDW is decremented at each successful read data transfer. The size of the address decrement depends on the Transfer Width.
'0'	X	'1'	SASHDW is incremented at each successful read data transfer. SASHDW is updated with the value stored in DMAn_SAi at the end of a block.
'1'	X	X	SASHDW remains constant.

Table 41-21. Destination Address Shadow update behaviour

DMAn_Di:FD	DMAn_Di:DED	DMAn_Di:FBD	Description of DMAn_DASHDWi:DASHDW update behaviour
'0'	'0'	'0'	DASHDW is incremented at each successful write data transfer. The size of the address increment depends on Transfer Width.
'0'	'1'	'0'	DASHDW is decremented at each successful write data transfer. The size of the address decrement depends on the Transfer Width.
'0'	X	'1'	DASHDW is incremented at each successful write data transfer. DASHDW is updated with the value stored in DMAn_DAi at the end of a block.
'1'	X	X	DASHDW remains constant.

Figure 41-19. Illustration of a DMAn_SASHDWi:SASHDW update



At the successful end of a DMA transfer, the Source Address or Destination Address can be updated with the value stored in Source Address Shadow or Destination Address Shadow respectively. This can be configured with the bits Update Source Address (DMAn_Di:US) or Update Destination Address (DMAn_Di:UD).

DMA transfer size

The DMA transfer size is calculated using the following formula:

DMA transfer size [byte] = Number of data transfers * ($2^{\text{Transfer Width}}$) =

= (DMAn_Ai:BC + 1) * (DMAn_Ai:TC + 1) * ($2^{\text{DMAn_Bi:TW}}$)

Transfer Count (DMAn_Ai:TC[15:0]) determines the number of blocks to be transferred in a DMA transfer. Block Count (DMAn_Ai:BC[3:0]) determines the number of data transfers in each block.

DMA transfer completion and error handling

Each DMA channel issues an interrupt at the end of a DMA transfer. This can be either a completion interrupt if the DMA transfer completed successfully or an error interrupt in case of an error condition or a stop request. A completion interrupt is signalled with the DIRQ flag (DMAn_Bi:DQ) and an error interrupt with the EDIRQ flag (DMAn_Bi:EQ). There is a DMA transfer end code associated with each interrupt, which is encoded in Stop Status (DMAn_Bi:SS[2:0]).

If a DMA transfer is completed successfully and the completion interrupt raised, Stop Status will show 'normal end' (DMAn_Bi:SS = '101'). If it is ended in error and the error interrupt raised, Stop Status will show one of the following possibilities:

- Stop request (DMAn_Bi:SS = '010')
- Source access error (DMAn_Bi:SS = '011')

- Destination access error (DMAn_Bi:SS = '100')

A 'stop request' during a running DMA transfer can be caused by asserting the stop request signal (DSTP) of the DMA transfer requesting client if the DMA channel is disabled (DMAn_Ai:EB), if the complete DMA Controller is disabled (DMAn_R:DE); or if a debug event occurs and the DMA Controller is configured to stop on a debug event.

Both interrupts, completion as well as error interrupt can be masked with the Completion Interrupt (DMAn_Bi:CI) and Error Interrupt (DMAn_Bi:EI) bits respectively. If these bits are set to '1' the interrupts are not masked. All unmasked completion interrupts are ORed and signalled to the Interrupt Controller. The same is done for the error interrupts.

All completion Interrupt Flags are in addition to the channel registers, available in two 32-bit registers (DMAn_DIRQ1 and DMAn_DIRQ2) for easier software handling. All Error Interrupts are handled in the same way and are available in registers DMAn_EDIRQ1 and DMAn_EDIRQ2.

The completion interrupt DMAn_Bi:DQ must be cleared by setting Clear DIRQ (DMAn_Ci:CD). Error interrupt EQ must be cleared by setting Clear EDIRQ (DMAn_Ci:CE). Stop Status will be cleared to its 'initial value' (DMAn_Bi:SS = '000') if DMAn_Bi:DQ or DMAn_Bi:EQ is set to '1'.

Source and destination protection

Each DMA channel has the possibility to define source and destination protection information independently. This information will be used by the AHB master and driven on the AHB protection control signals (HPROTM[3:0]) for use by peripherals which have implemented access protection as defined by the AHB protocol. Protection information must be provided by the Source Protection (DMAn_Bi:SP[3:0]) or Destination Protection (DMAn_Bi:DP[3:0]) bits if used. The DMA Controller only does data transfers; DMAn_Bi:SP[0] and DMAn_Bi:DP[0] are statically set to '1'. The initial value indicates a 'Not cacheable/Not bufferable/Privileged access/Data access' source and destination protection.

Channel disabling and halting

After a reset, a DMA channel is disabled by default in order to properly configure it before a DMA request is serviced. A DMA channel is enabled by setting Channel Enable (DMAn_Ai:EB) to '1', after which the channel waits for a DMA request.

Each DMA channel can be independently disabled. This is done by setting Channel Enable (DMAn_Ai:EB) to '0'. Setting this bit to '0' can be done at any time but has different effects when it is done. If DMAn_Ai:EB is set to '0' during a running DMA transfer, the transfer is stopped at the next transfer gap, an error interrupt is raised and the channel is disabled. The transfer gap means the DMA Controller has transferred a block of data and the AHB master interface releases the bus request for a few cycles. When DMAn_Ai:EB is set to '0' while an interrupt is pending (DMAn_Bi:DQ = '1' or DMAn_Bi:EQ = '1') or no DMA transfer is running, there will be no other effects besides the channel is disabled.

Channel halting is done by setting the Pause Bit (DMAn_Ai:PB) to '1'. If this bit is set to '1' during a running DMA transfer, it will halt after completion of the current transferred block. If it is set to '1' before receiving a transfer request the halt state is entered immediately. Clearing this bit will put the channel into run state and it will wait for the next transfer request to continue the DMA transfer or if a transfer request is already pending, it will continue immediately.

41.3.1.3 Burst transfer mode

Burst transfer mode is almost identical to block transfer mode. The only difference is the DMA transfer request. Whereas in block transfer mode one request for each block of data is needed, in burst transfer mode only one request is needed at the beginning of a DMA transfer for the complete transfer. The requests needed for subsequent blocks of data is generated internally by the DMA Controller itself.

41.3.1.4 Demand transfer mode

In demand transfer mode, the DMA client requests data to be transferred as long as the transfer request is asserted. However the length of a transfer is limited to the 'Block Count' number of data transfers during an arbitration phase. The DMA client can control by asserting or deasserting the request signal how data is transferred over time. The DMA transfer will be successfully completed if the specified number of data transfers are done without error or unsuccessfully completed if an error condition occurred.

After each transferred block of data, the DMA arbiter does another arbitration and proceeds with the next requesting channel with the highest priority. The arbitration depends on the selected arbitration scheme and the set priorities of the requesting channels (for details about the arbitration see 41.3.3 DMA arbiter).

DMA transfer requests

In demand transfer mode only hardware requests are possible.

For a channel hardware request, the Input Select bits (DMAi_An:IS) need to be set to '01'. The DMA client which is routed through the DMA Client Matrix to the channel will give the trigger by asserting DREQ. Refer to 41.3.2 DMA Client Matrix for the function and configuration of the DMA Client Matrix and to 41.3.5 External DMA Client Interface signal requirements for the requirements of DREQ for an external DMA client.

The DMA client needs to assert DREQ until the DMA Controller acknowledges this request by asserting DREQ_ACK. From this point in time data is transferred as long as DREQ is asserted. The maximum number of data transfers which can be done is determined by Block Count (DMAn_Ai:BC[3:0]). The DMA client can also deassert its request if no data transfer is wanted. In order not to block the DMA Controller for too long by deasserting DREQ, this time is limited.

The DMA client can de-assert DREQ for Timeout (DMAn_Ai:TO[3:0]) cycles (DMA Controller clock cycles) continuously in order not to get a time-out. When DREQ is asserted again before time-out has been reached, time-out is reset to the value set in DMAn_Ai:TO. When time-out has been reached the arbitration phase for this channel is ended and DREQ_ACK is deasserted. The arbiter then continues with the next requesting channel with the highest priority. Software requests are not available in demand transfer mode.

Block of data

A block of data is determined by the Block Count (DMAn_Ai:BC[3:0]) and Transfer Width (DMAn_Bi:TW[1:0]) bit settings. The DMA Controller can make a maximum of DMAn_Ai:BC + 1 data transfers from the source address range starting at Source Address (DMAi_SAn:SA) to the destination address range starting at Destination Address (DMAn_DAi:DA). The DMA client controls how many data transfers will be made during the arbitration phase. The AHB master interface issues only alternating SINGLE reads and SINGLE writes. See Table 41-19 with settings DMAn_Ai:BC = 0-16, DMAn_Ai:BL = SINGLE, and DMAn_Ai:AL = '1' for the possible AHB transfers.

Addressing in demand transfer mode is equal to addressing in block transfer mode.

DMA transfer size

The DMA transfer size is calculated by the following formula.

DMA transfer size [Byte]= Number of data transfers * ($2^{\text{Transfer Width}}$) =

= (DMAn_Ai:TC + 1) * ($2^{\text{DMAn_Bi:TW}}$)

Transfer Count (DMAn_Ai:TC[15:0]) determines the number of data transfers to be done in a DMA transfer.

DMA transfer completion and error handling

DMA transfer completion and error handling in demand transfer mode is the same as that described in block transfer mode (See DMA transfer completion and error handling).

Source and destination protection

Source and destination protection in demand transfer mode is the same as that described in block transfer mode (See Source and destination protection).

Channel disabling and halting

Channel disabling and halting in demand transfer mode is the same as that described in block transfer mode (See Channel disabling and halting).

41.3.2 DMA Client Matrix

41.3.2.1 Overview

The DMA Client Matrix provides the possibility to route 'M' DMA clients to 'I' DMA channels. 'M' is greater than or equal to 'I'. The selection of which DMA channel serves which DMA client will be set with the Client Interface (DMAn_CMCHICn:CI). Out of 'M' DMA client sources are 'K' client interfaces which serve external DMA clients. The configuration of the external DMA

Client Interfaces will be done with the DMAn_CMECICk Register. The configuration of the internal DMA clients will be done with the registers DMAn_CMICICj.

41.3.2.2 Modes of operation

Each DMA Client Interface can work in one of the following modes:

1. Disabled mode

   External DMA clients:

   In this mode the External DMA Client Interface is disabled. The enabling and disabling is done via the DMA Client Matrix Client Configuration Register (DMAn_CMECICk:ENCI). Enabling can take place when the DMA channel is configured and enabled. Disabling shall take place after the channel has completed a DMA transfer and before its interrupt (DMAn_Ci:CD or DMAn_Ci:CE) is cleared. Reconfiguration of the external DMA Client Interface shall only be done in disabled mode.

   Internal DMA clients:

   An Internal DMA Client Interface is disabled if it is not selected by any of the DMA Client Matrix Channel Configuration Registers (DMAn_CMCHICi:CI). Reconfiguration of the internal DMA Client Interface shall only be done in disabled mode.

2. Normal mode

   In this mode a DMA channel is routed directly to the specified (DMAn_CMCHICi:CI) DMA client. The operation of the DMA Client Matrix in this mode is fully transparent and behaves as if the DMA client would be connected directly to the DMA Channel Interface.

41.3.2.3 Functional description

Purpose of the DMA Client Matrix is to provide flexibility in the use of the available DMA channels. The configuration of the DMA Client Matrix is intended to be static and shall be done after the boot code execution when the software is setting up the system. However, the DMA Client Matrix configuration can be changed during normal operation of the system if the procedure described in the following paragraphs is followed:

- Disabling an external DMA Client Interface
- Enabling and configuring an external DMA Client Interface

Structure of the DMA Client Matrix

The DMA Client Matrix will be a full matrix where each DMA Client Interface 'm' can be routed to every DMA Channel Interface 'i'.

DMA Client Matrix configuration

The configuration of the DMA Client Matrix is done with the following registers:

- DMA Controller Client Matrix External Client Configuration Registers (DMAn_CMECICk)
- DMA Controller Client Matrix Internal Client Configuration Registers (DMAn_CMICICj)
- DMA Controller Client Matrix Channel Interface Configuration Registers (DMAn_CMCHICi)

The DMA Controller Client Matrix External Client Configuration Register contains three enable bits, six Signal Level Select bits and three Signal Behaviour bits. For their function see the descriptions below:

The Config bit Enable Client Interface, DMAn_CMECICk:ENCI, enables the external DMA Client Interface 'k'. This means that DREQ[k] and DREQ_ACK[k] of DMA Client Interface 'k' is routed without interaction to DMA Channel Interface 'i', if selected in DMAn_CMCHICi:CI. The configuration of the DMA Client Interface must take place before it is enabled. Disabling of the interface must take place when a transfer has finished and before the Interrupt flag, either DMAn_Bi:DQ or DMAn_Bi:EQ, is cleared by setting DMAn_Ci:CD or DMAn_Ci:CE. By default the DMA Client Interface is disabled. If DMAn_CMECICk:ENCI is set to '0', the output signals of external DMA Client Interface 'k' (DREQ_ACK[k], DSTP_ACK[k], DEOP[k]) are set to inactive level and the interface does not react to signal changes on its input signals (DREQ[k], DSTP[k], DEOP_ACK[k]) or the request signals are connected to the associated acknowledge signals (DREQ[k] <-> DREQ_ACK[k], DSTP[k] <-> DSTP_ACK[k]. These two ways of behaviour can be chosen by setting of the respective Behaviour Config bits (DMAn_CMECICk:BEHREQACK and DMAn_CMECICk:BEHSTPACK) in register DMAn_CMECICk. Connecting a request signal to its associated acknowledge signal can be used to reset a falsely set DMA request signal without violating the two-way handshake protocol.

The Config bit Enable Stop, DMAn_CMECICk:ENSTP, enables the use of the external DMA Client Interface signal pair DSTP[k]/DSTP_ACK[k] if the external DMA Client Interface is enabled by DMAn_CMECICk:ENCI. If disabled the DMA Client Matrix sets DSTP_ACK[k] to inactive level and does not react on signal changes on DSTP[k] or DSTP[k] is connected to DSTP_ACK[k]. These two ways of behaviour can be chosen by setting of the Behaviour Config bit, DMAn_CMECICk:BEHSTPACK, in register DMAn_CMECICk. The setting of DMAn_CMECICk:ENSTP must take place before DMAn_CMECICk:ENCI is set to '1'. By default DMAn_CMECICk:ENSTP is set to '0' and thus disables the use of DSTP[k]/DSTP_ACK[k]. The setting of DMAn_CMECICk:ENSTP has no effect as long as DMAn_CMECICk:ENCI is set to '0'. The Config bit Enable End of Processing, DMAn_CMECICk:ENEOP, enables the use of the external DMA Client Interface signal pair DEOP[k]/DEOP_ACK[k] if the external DMA Client Interface 'k' is enabled by DMAn_CMECICk:ENCI. If disabled the DMA Client Matrix sets DEOP[k] to the inactive level and does not react to signal changes on DEOP_ACK[k]. The setting of DMAn_CMECICk:ENEOP must take place before DMAn_CMECICk:ENCI is set to '1'. By default DMAn_CMECICk:ENEOP is set to '0' and thus disables the use of DEOP[k]/DEOP_ACK[k]. The setting of DMAn_CMECICk:ENEOP has no effect as long as EDMAn_CMECICk:ENCI is set to '0'. Furthermore, the DMA Client Matrix provides the user with the flexibility to configure signal levels of the six external DMA Client Interface signals to match the signal levels of an external DMA client. For each signal of the external DMA Client Interface exists one Configuration bit to set the signal level to either high-active or low-active. The default selected level is high-active. See Table 41-16 for the complete description of the Signal Level Configuration bits.

The Config bit Behaviour Request Acknowledge, DMAn_CMECICk:BEHREQACK, sets the behaviour of the output signal DREQ_ACK[k] if the external DMA Client Interface 'k' is not selected in any of the Channel Configuration Registers (DMAn_CMCHICi:CI) or the External DMA Client Interface 'k' is disabled (DMAn_CMECICk:ENCI = '0'). The user can choose that DREQ_ACK[k] drives inactive level or that DREQ[k] is connected to DREQ_ACK[k] in that case. The latter can be used to reset a falsely set DMA request signal (due to software misbehaviour) without violating the handshake protocol.

The Config bit Behaviour Stop Acknowledge, DMAn_CMECICk:BEHSTPACK, sets the behaviour of the output signal DSTP_ACK[k] if the external DMA Client Interface is not selected in any of the Channel Configuration Registers (DMAn_CMCHICi:CI); or the Enable Stop (DMAn_CMECICk:ENSTP) is set to '0'; or the external DMA Client Interface 'k' is disabled (DMAn_CMECICk:ENCI = '0'). The user can choose that DSTP_ACK[k] drives the inactive level or that DSTP[k] is connected to DSTP_ACK[k] in that case. The later can be used to reset a falsely set DMA stop request signal without violating the two-way handshake protocol.

The DMA Controller Client Matrix Internal Client Interface Configuration Register contains two signal behaviour bits:

1. **The Config bit Behaviour Request Acknowledge (DMAn_CMICICj:BEHREQACK)** sets the behaviour of the output signal DREQ_ACK[j] if the internal DMA Client Interface 'j' is not selected in any of the Channel Configuration Registers (DMAn_CMCHICi). The user can choose that DREQ_ACK[j] drives inactive level or that DREQ[j] is connected to DREQ_ACK[j] in that case. The latter can be used to reset a falsely set DMA request signal (due to software misbehaviour) without violating the two-way handshake protocol.
2. **The Config bit Behaviour Stop Acknowledge (DMAi_CMICICj:BEHSTPACK)** sets the behaviour of the output signal DSTP_ACK[j] if the internal DMA Client Interface 'j' is not selected in any of the channel Configuration Registers (DMAn_CMCHICi:CI). The user can choose that DSTP_ACK[j] drives inactive level or that DSTP[j] is connected to DSTP_ACK[j] in that case.

The DMAC Client Matrix Channel Interface Configuration Register contains up to nine selection bits:

The Selection bits Client Interface, DMAn_CMCHICi:CI, specifies which DMA Client Interface 'm' is connected to the DMA Channel Interface 'n'. The configuration of these bits must take place before DMAn_R:DE and DMAn_Ai:EB is set to '1'. The client interface number must be programmed as the binary value to DMAn_CMCHICi:CI. The setting of CI makes the connection between the DMA Client Interface (defined by the value of CI) and the DMA Channel Interface 'i'. Selecting the same DMA Client Interface in any of the DMA Client Matrix Channel Configuration Registers two or more times results in unpredictable behaviour of the DMA Controller and must be avoided.

The availability of certain DMA clients depends on the specific device. Refer to the device-specific datasheet for the availability of DMA clients.

41.3.2.4 Initialization and application information

Reset

The reset state of each DMA Client Matrix Configuration bit is shown in the register description of DMAn_CMECICk, DMAn_CMICICj and DMAn_CMCHICi. To summarize, after a hardware reset, all external and internal DMA Client Interfaces

are disabled, all signals are set to high-active level, and DMA Channel Interface 0 - 'I-1' is configured to route to the DMA Client Interface 0 - 'I-1'.

Note: Selecting the same DMA Client Interface in two or more DMA Channel Interfaces leads to unpredictable behaviour of the DMA Controller. Therefore DMAn_CMCHICi:CI must be properly configured before enabling the DMA Controller and one or more of its channels.

Enabling and configuring an external

DMA Client Interface To enable an external DMA Client Interface the steps listed below must be followed:

1. Determine which DMA Channel Interface the DMA Client Interface will be connected to.
2. Determine if the DMA signal pair DSTP/DSTP_ACK and/or DEOP/DEOP_ACK will be used.
3. Determine the signal level of all used external DMA Client Interface signals.
4. Set the Configuration bits in DMAn_CMECICk according to the outcome of points 2 and 3. Do not enable the client interface yet.
5. Ensure that the DMA Channel Interface the DMA client connects to is properly configured. In addition, the proper Port Pin configuration has to be done for MCU external DMA clients (refer to 18. Port Pin Configuration). Enable the DMA channel and the DMA Controller if not done already.
6. Enable the DMA Client Interface by setting DMAn_CMECICk:ENCI to '1'.

Disabling an external DMA Client Interface

To disable an external DMA Client Interface the Enable Client Interface bit, DMAn_CMECICk:ENCI, has to be cleared. Clearing this bit shall be done only while the DIRQ (DMAn_Bi:DQ) or EDIRQ (DMAn_Bi:EQ) flags are asserted or the connected DMA channel is disabled. This situation can be reached in the following ways:

1. The DMA transfer on the interface has already been successfully completed (--> DMAn_Bi:DQ is set) or has generated an error (--> DMAn_Bi:EQ is set). The interrupt service routine for the selected DMA Channel Interface must then clear DMAn_CMECICk:ENCI.
2. The DMA transfer is in the middle of the transfer or has not yet started. Clear the DMA Channel Enable bit DMAn_Ai:EB of the connected DMA channel. In case of a running DMA transfer this will be treated as a stop request, the running transfer is stopped at the next transfer gap and DMAn_Bi:EQ will be set. Otherwise, if the transfer has not yet started, clearing DMAn_Ai:EB will not lead to the setting of DMAn_Bi:BQ or DMAn_Bi:DQ. Now the DMA Client Interface can be disabled by clearing DMAn_CMECICk:ENCI.

41.3.3 DMA arbiter

41.3.3.1 Overview

The DMA arbiter is responsible for choosing a DMA channel based on the arbitration scheme selected in the Global Configuration Register (DMAn_R:PR). There are three arbitration schemes available:

- Fixed priority
- Dynamic priority
- Round-robin

The arbitration schemes are explained in detail in the following sections. The arbitration scheme can be changed at any time; however, it becomes effective only after the current running data transfer has been completed at the next transfer gap.

41.3.3.2 Fixed priority

In fixed priority arbitration scheme the DMA channels have a 'fixed' priority which can be set with DMAn_Bi:PN. A Priority Number (PN) equal to '0' has the highest priority, whereas a PN equal to 127 has the lowest priority. DMA channels with equal PN, the channel with the lowest channel number 'i' has the highest priority. The initial value of DMAn_Bi:PN is 127. Priority Number can be changed at any time; however, it becomes effective only after the current running data transfer has been completed at the next transfer gap. To illustrate this behaviour, Table 41-22 shows an arbitration example for eight DMA channels.

Table 41-22. Fixed priority arbitration example

Arbitration cycle	Requesting DMA channel 'i'	PN of requesting DMA channel 'i'	Grant given to DMA channel 'i'
x+1	2	0	2
	4	2	
	7	6	
x+2	4	2	4
	7	6	
	8	5	
x+3	4	2	4
	7	6	
	8	5	
x+4	1	5	1
	7	6	
	8	5	

Table 41-23. Fixed priority arbitration example

Arbitration cycle	Requesting DMA channel 'i'	PN of requesting DMA channel 'i'	Grant given to DMA channel 'i'
x+5	3	9	8
	7	6	
	8	5	

41.3.3.3 Dynamic priority

The dynamic priority arbitration scheme is an extension of the fixed priority arbitration scheme. The priority of the DMA channels is dynamically adjusted based on the criterion of whether a channel request has been granted or not. If a channel request has been granted, its dynamic PN is loaded with the PN stored in DMAn_Bi:PN and if a channels request was not granted, its dynamic PN is decremented by '1'. The arbiter grants permission to the requesting DMA channel with the lowest dynamic PN. If two or more requesting channels have equal dynamic PN, the DMA channel with the lowest channel number 'i' has the highest priority and will win the arbitration process. A PN can be changed any time; however, it becomes effective after it has been re-loaded by the channel, i.e. after the channel has won the arbitration. To illustrate this behaviour, Table 41-24 shows an arbitration example for four DMA channels.

Note:

In case a channel's re-programmed PN should become effective immediately (not only after the re- programmed channel has won the arbitration), this behaviour can be forced by changing the arbitration scheme to Fixed priority and then changing it

back to dynamic priority while in halt state. However, it must be noted that the dynamic priority of all channels will be re-loaded with the value contained in DMAi_Bn:PN and not only the re-programmed channels.

Table 41-24. Dynamic priority arbitration example

Arbitration cycle	Requesting DMA channel		Dynamic PN of DMA channel	PN of DMA channel	Grant given to DMA channel
1	Ch. 0	Yes	1	1	0
	Ch. 1	Yes	2	2	
	Ch. 2	Yes	3	3	
	Ch. 3	No	3	3	
2	Ch. 0	No	1	1	1
	Ch. 1	Yes	1	2	
	Ch. 2	Yes	2	3	
	Ch. 3	No	3	3	
3	Ch. 0	No	1	1	2
	Ch. 1	No	2	2	
	Ch. 2	Yes	1	3	
	Ch. 3	Yes	3	3	
4	Ch. 0	No	1	1	1
	Ch. 1	Yes	2	2	
	Ch. 2	No	3	3	
	Ch. 3	Yes	2	3	
5	Ch. 0	No	1	1	3
	Ch. 1	Yes	2	2	
	Ch. 2	Yes	3	3	
	Ch. 3	Yes	1	3	

41.3.3.4 Round-robin

In the round-robin arbitration scheme the turn is rotated in directional and cyclic order from DMA channel 0 to DMA channel 'n'. At most one DMA channel request can be granted at any time, and this is defined as a turn being given. The turn is moved forward at each transfer gap. The turn's rotation is not strictly round-robin in order not to waste an arbitration phase by giving the turn to a non-requesting DMA channel. Instead the turn is given to the next requesting DMA channel in the rotation direction. If no DMA channel was last served (only possible in initial state) the requesting DMA channel with the lowest channel number 'n' has the highest priority and will win the arbitration process. To illustrate this behaviour, Table 41-25 shows an arbitration example for eight DMA channels.

Table 41-25. Round-robin arbitration example

Arbitration cycle	Requesting DMA channels	Grant given to DMA channel	Last served DMA channel
1	2	2	None
	4		
	7		
2	4	4	2
	7		
	8		
3	4	7	4
	7		
	8		
4	1	8	7
	4		
	8		
5	1	1	8
	4		
	7		

41.3.3.5 Application information

Fixed priority arbitration

With this arbitration scheme the DMA channel request from the channel with the highest priority will be selected for service. If the DMA Controller is programmed that channel 0 is assigned the highest priority and this channel has a higher service request rate compared to the other channels, it is possible that this channel absorbs the complete bandwidth of the DMA Controller, meaning that the other channels will not be serviced.

Dynamic priority arbitration

With this arbitration scheme starving can be avoided by assigning a requesting channel, which was not granted permission, to the next higher priority level. However, starving cannot be avoided if the DMA Controller is not programmed properly i.e. if channel 0 is assigned the highest priority the other channels cannot reach a higher priority than channel 0. If this channel has in addition a higher service request rate that the other channels, it will use the complete bandwidth of the DMA Controller.

Round-robin arbitration

With this arbitration scheme, starving the requesting channels is not possible even if channel 0 has a service request rate which is equal to or exceeds the arbitration rate.

41.3.4 DMA AHB slave interface

This is the DMA Controller’s system interface through which the DMA Controller registers are accessed.

41.3.4.1 Supported data transfers

The slave interface supports 8-, 16- and 32-bit wide AHB data transfers. 16-bit and 32-bit accesses shall be 16- and 32-bit address aligned respectively.

Single data and fixed incremental burst accesses are supported (SINGLE, INCR4, INCR8, and INCR16).

Note: All accesses to DMA Controller registers must be done in privileged mode.

41.3.4.2 Data transfer response

The DMA AHB slave interface responds with the following possibilities to any kind of access:

- OKAY response
- ERROR response

The ERROR response will be given for accesses where a protection error or a register access error occurs.

Protection error

A protection error is raised if one of the conditions listed in the following table is met.

Table 41-26. Protection error conditions

HWRITES	HPROTS		Description	Protection error
	[1]	[0]		
0	0	0	read/user access/opcode fetch	1
0	1	0	read/privileged access/opcode fetch	1

Register access error

A register access error is raised if a read or write to a reserved address location is attempted. For the location of the reserved addresses see Table 41-1.

41.3.5 External DMA Client Interface signal requirements

An external DMA Client Interface can consist of up to three request/acknowledge signal pairs. This depends on the setting of DMAn_CMECICk:ENCI, DMAn_CMECICk:ENSTP or DMAn_CMECICk:ENEOP. The three signal pairs are:

- DMA transfer request/DMA transfer request acknowledge (DREQ/DREQ_ACK)
- DMA stop request/DMA stop request acknowledge (DSTP/DSTP_ACK)
- DMA 'end of processing'/DMA 'end of processing' acknowledge (DEOP/DEOP_ACK)

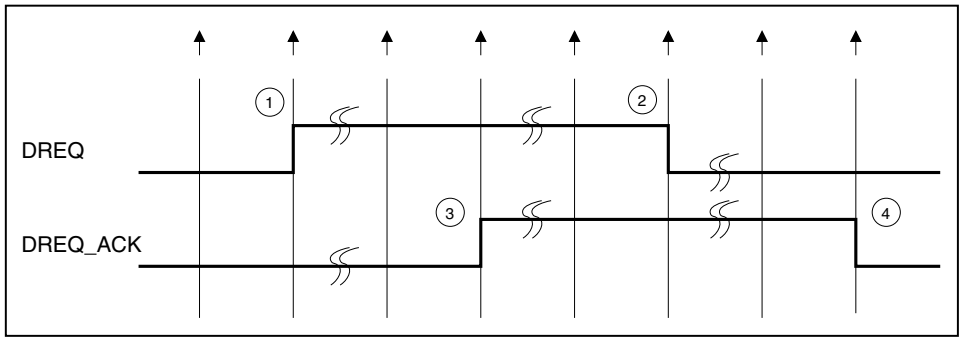
Block and burst transfer mode

Each request/acknowledge signal pair follows a two-way handshake protocol. The signal protocol must follow the rules below (DREQ/DREQ_ACK exemplarily):

1. DREQ shall only be asserted if DREQ_ACK is de-asserted.
2. DREQ shall only be de-asserted if DREQ_ACK is asserted.
3. DREQ_ACK will only be asserted if DREQ is asserted.
4. DREQ_ACK will only be de-asserted if DREQ is de-asserted.

Figure 41-20 illustrates the rules with high-level active signals for DREQ and DREQ_ACK.

Figure 41-20. Two-way handshake protocol DREQ/DREQ_ACK (block/burst transfer mode)



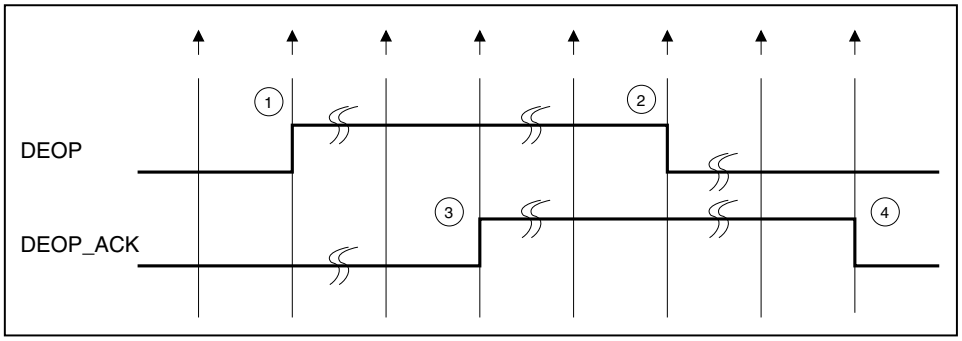
The protocol rules of signal pair DSTP/DSTP_ACK are equivalent to the ones for DREQ/DREQ_ACK.

The signal protocol for DEOP/DEOP_ACK must follow the rules below:

1. DEOP will only be asserted if DEOP_ACK is de-asserted.
2. DEOP will only be de-asserted if DEOP_ACK is asserted.
3. DEOP_ACK shall only be asserted if DEOP is asserted.
4. DEOP_ACK shall only be de-asserted if DEOP is de-asserted

Figure 41-21 illustrates the rules with high-level active signals for DEOP and DEOP_ACK.

Figure 41-21. Two-way handshake protocol DEOP/DEOP_ACK (block/burst transfer mode)



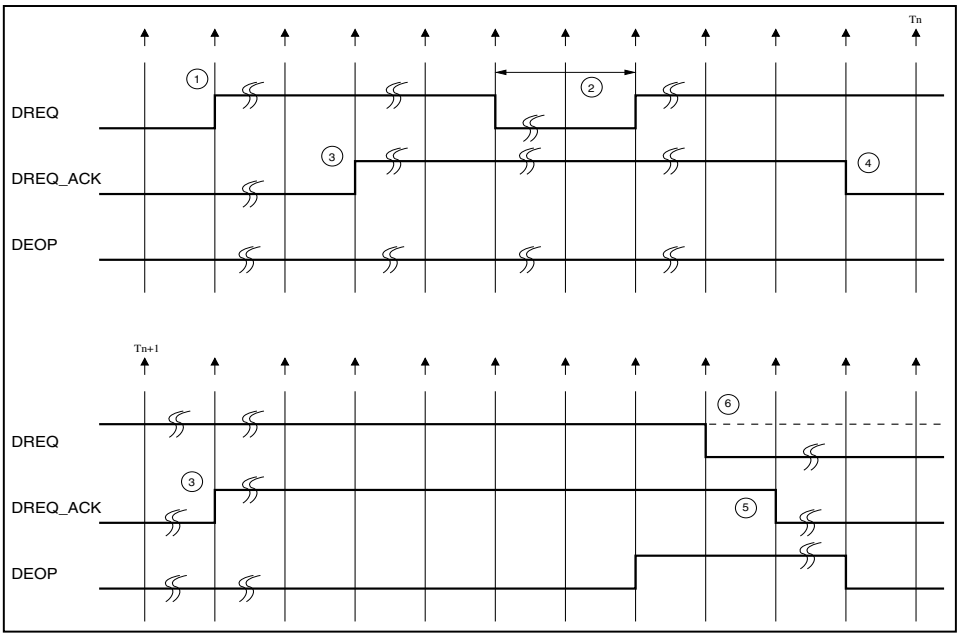
Demand transfer mode

In demand transfer mode the two-way handshake protocol is slightly different than in block or burst transfer mode. The transfer request/acknowledge protocol must follow the rules below:

1. DREQ can be asserted any time.
2. DREQ should only be deasserted for less than the Timeout (DMAn_Ai:TO) cycles while DREQ_ACK is asserted. Otherwise the arbitration phase will be terminated prematurely (before BC + 1 data transfers are done).
3. DREQ_ACK will only be asserted if DREQ is asserted.
4. DREQ_ACK will be de-asserted if DREQ is de-asserted for TO cycles or if DMAn_Ai:BC + 1 data transfers were done.
5. DREQ_ACK will be de-asserted if DREQ is de-asserted for TO cycles or if DMAn_Ai:TC + 1 data transfers were done.
6. DREQ can be de-asserted when DEOP is asserted and DEOP is used by the DMA Client Interface. In case the DMA Client Interface does not use signal pair DEOP/DEOP_ACK it can de-assert DREQ at any time or just leave it asserted. If it is left asserted it will be recognized as new DMA transfer request as soon as the completion interrupt is cleared.

Figure 41-22 illustrates the rules with high-level active signals for DREQ and DREQ_ACK.

Figure 41-22. Two-way handshake protocol DREQ/DREQ_ACK (demand transfer mode)



Protocol rules for DSTP/DSTP_ACK and DEOP/DEOP_ACK are in demand transfer mode are the same as in block/burst transfer mode.

Areas of application

Data transfers from every location to every location except:

- The System Controller
- Watchdog
- Debug bus
- DMA Controller MPU
- DMA Controller registers

Table 41-27 shows an example of client numbering for J = 16, K = 4, and M =24.

Table 41-27. DMA client numbering example

Client number 'm'	External client number 'k'	Internal client number 'j'	DMA client
0	0		MCU external DMA client 0
1	1		MCU external DMA client 1
2	2		MCU external DMA client 2
3	3 = K - 1		MCU external DMA client 3
4			Not available
5			Not available
6			Not available
7			Not available

Table 41-27. DMA client numbering example

Client number 'm'	External client number 'k'	Internal client number 'j'	DMA client
8		0	MCU internal DMA client 0
9		1	MCU internal DMA client 1
10		2	MCU internal DMA client 2
11		3	MCU internal DMA client 3
12		4	Not available
13		5	Not available
14		6	MCU internal DMA client 6
15		7	MCU internal DMA client 7
16		8	MCU internal DMA client 8
17		9	MCU internal DMA client 9
18		10	Not available
19		11	MCU internal DMA client 11
20		12	MCU internal DMA client 12
21		13	MCU internal DMA client 13
22		14	MCU internal DMA client 14
23 = M - 1		15 = J -1	MCU internal DMA client 15

Notes:

- N: Number of DMA Controller instances.
  n = 0 ...N-1
- J: Number of MCU internal DMA Client Interfaces.
  J = M - 8. j = 0 ...J-1
- K: Number of MCU external DMA Client Interfaces.
  k = 0 ...K-1
- M: Number of DMA Client Interfaces.
  m = 0 ...M-1
- I: Number of DMA channels.
  i = 0 ...I-1
- For the actual numbers of I, K, M, and N refer to the device-specific datasheet.

42. External Bus Interface

This chapter explains the function and operation of the External Bus Interface (EBI).

42.1 Outline of the External Bus Interface

This section describes the features and the block diagram of the External Bus Interface (EBI).

The EBI is a generic memory controller, which can access external memories like SRAM/FLASH or SDRAM. It can support access to eight SRAM/FLASH and one SDRAM memory.

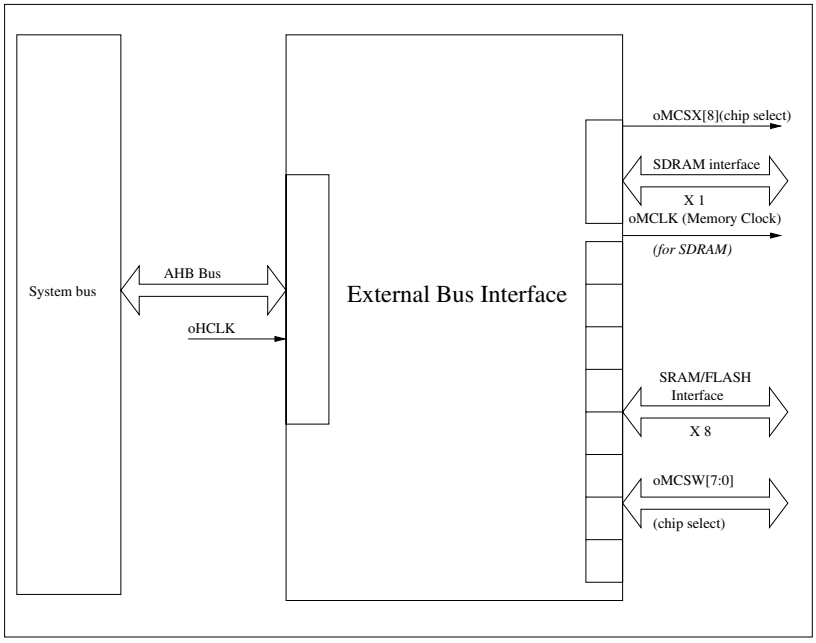
Features of External Bus Interface

- Supports 8-, 16- and 32-bit width SRAM/FLASH
- Supports 16- and 32-bit SDRAM bus widths
- Supports bi-endian access for SRAM/FLASH
- Eight chip selects for SRAM, 1 chip select for SDRAM
- The device can be accessed during NAND FLASH accessing (no need for exclusive access control)
- Supports NOR FLASH page accesses
- Supports SDRAM power-down mode (automatic setting and interval setting are available)

Block diagram of the EBI

The Figure 42-1 shows the block diagram of the EBI.

Figure 42-1. Block diagram of the EBI



42.1.1 Pad interface

Table 42-1. External I/F pins of the EBI module

Name	IO	Clock domain	Active	Default (Steady State)	Description
oMAD[24:0]	O	oMCLK	N/A	N/A	Address output. Lower 16-bit are shared by SDRAM/ SRAM.
oMDQM[3:0]	O	oMCLK	L	4'b1111	Byte mask signal for SRAM.
oMWEX	O	oMCLK	L	H	Write enable signal for SRAM.
oMOEX	O	oMCLK	L	H	Read enable for SRAM.
oMCLK	O	oHCLK	clock	clock	Clock for SDRAM. Gated version of oHCLK.
oMCSX[8:0]	O	oMCLK	L	9'h1F	Chip select. SRAM/FLASH: oMCSX[7:0] SDRAM : oMCSX[8]
oMCASX	O	oMCLK	L	H	SDRAM: Column Address Select.
oMRASX	O	oMCLK	L	H	SDRAM: Row Address Select.
oMDWEX	O	oMCLK	L	H	SDRAM: Write Enable.
oMCKE	O	oMCLK	H	H	SDRAM: Clock Enable.
oMNALE	O	oMCLK	H	L	NAND Flash: Address Latch Enable.
oMNCLE	O	oMCLK	H	L	NAND Flash: Command Latch Enable.
oMNWEX	O	oMCLK	L	H	NAND Flash: Write Enable.
oMNREX	O	oMCLK	L	H	NAND Flash: Read Enable.
iMRDY	I	oMCLK	N/A	N/A	Memory Ready.
iMRDatb[31:0]	I	oMCLK	N/A	N/A	Read Data.
oMWDatb[31:0]	O	oMCLK	N/A	N/A	Write Data.

Note:

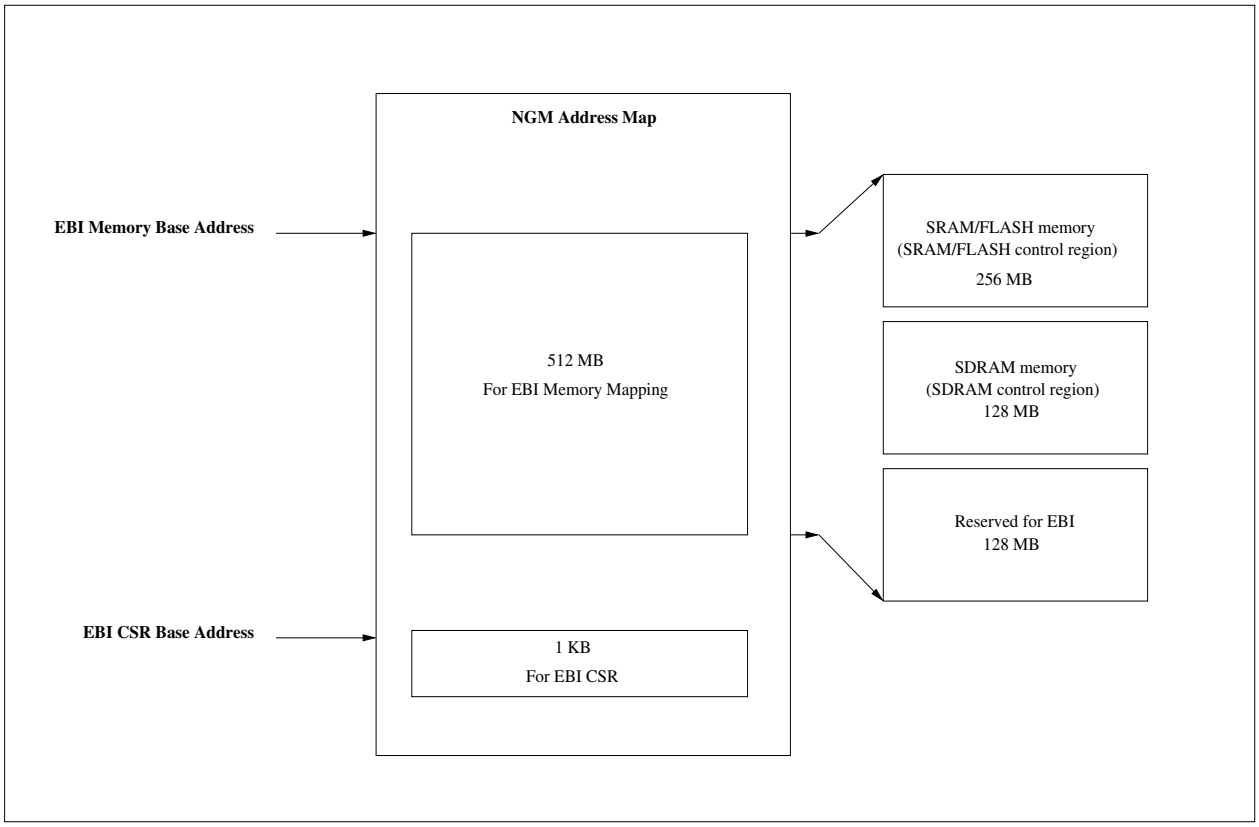
oMCLK - Memory clock

oHCLK - Configuration clock.

42.1.2 External Bus Interface and Memory regions

The EBI is configured by the setting register region. The control region is divided into the SRAM control region and the SDRAM control region.

Figure 42-2. EBI Memory region (Configuration register and control region)



Note: For details on EBI Memory regions refer to the device-specific datasheet.

42.2 External Bus Interface Configuration registers

This section lists all the EBI registers and explains their function in detail.

EBI configuration registers can be written with 32-bit accesses only. Write accesses with a size smaller than 32-bit return an error response. EBI configuration registers can be accessed only with correct PPU access settings.

Registers of EBI

- Unlock Register (EBI_UNLOCK)
- Lock Status Register (EBI_LSTSR)
- SRAM/FLASH Mode Control Register (EBI_SFMR0~7)
- SRAM/FLASH Access Configuration Register (EBI_SFACCR0~7)
- SRAM/FLASH Address Control Register (EBI_SFADDCR0~7)
- SDRAM Mode Control Register (EBI_SDMODCR)
- SDRAM Refresh Control Register (EBI_SDRCR)
- SDRAM Power Control Register (EBI_SDPCR)
- SDRAM Timing Control Register (EBI_SDTCR)
- SDRAM Command Register (EBI_SDCOMDR)
- Error Register (EBI_ERRR)

Table 42-2. Memory layout of External Bus Interface registers

Offset	+3	+2	+1	+0
0x00000000	EBI_UNLOCK 00000000 00000000 00000000 00000000			
0x00000004	EBI_LSTSR 00000000 00000000 00000000 00000001			
0x00000008	EBI_SFMR0 00000000 00000000 00000000 00000000			
0x0000000C	EBI_SFMR1 00000000 00000000 00000000 00000000			
0x00000010	EBI_SFMR2 00000000 00000000 00000000 00000000			
0x00000014	EBI_SFMR3 00000000 00000000 00000000 00000000			
0x00000018	EBI_SFMR4 00000000 00000000 00000000 00000000			
0x0000001C	EBI_SFMR5 00000000 00000000 00000000 00000000			
0x00000020	EBI_SFMR6 00000000 00000000 00000000 00000000			
0x00000024	EBI_SFMR7 00000000 00000000 00000000 00000000			

Table 42-2. Memory layout of External Bus Interface registers

Offset	+3	+2	+1	+0
0x00000028	EBI_SFACCR0 00000101 01011111 11110000 00001111			
0x0000002C	EBI_SFACCR1 00000101 01011111 11110000 00001111			
0x00000030	EBI_SFACCR2 00000101 01011111 11110000 00001111			
0x00000034	EBI_SFACCR3 00000101 01011111 11110000 00001111			
0x00000038	EBI_SFACCR4 00000101 01011111 11110000 00001111			
0x0000003C	EBI_SFACCR5 00000101 01011111 11110000 00001111			
0x00000040	EBI_SFACCR6 00000101 01011111 11110000 00001111			
0x00000044	EBI_SFACCR7 00000101 01011111 11110000 00001111			
0x00000048	EBI_SFADDCR0 00000000 00001111 00000000 00000000			
0x0000004C	EBI_SFADDCR1 00000000 00001111 00000000 00010000			
0x00000050	EBI_SFADDCR2 00000000 00001111 00000000 00100000			
0x00000054	EBI_SFADDCR3 00000000 00001111 00000000 00110000			
0x00000058	EBI_SFADDCR4 00000000 00001111 00000000 01000000			
0x0000005C	EBI_SFADDCR5 00000000 00001111 00000000 01010000			
0x00000060	EBI_SFADDCR6 00000000 00001111 00000000 01100000			
0x00000064	EBI_SFADDCR7 00000000 00001111 00000000 01110000			
0x00000068	EBI_SDMODCR 00000000 00000000 00010011 00000000			
0x0000006C	EBI_SDRCR 00000000 00000000 00000000 00101000			
0x00000070	EBI_SDPCR 00000000 00000000 00000000 00000000			

Table 42-2. Memory layout of External Bus Interface registers

Offset	+3	+2	+1	+0
0x00000074	EBI_SDTCR 00000000 01000010 00010001 01000001			
0x00000078	EBI_SDCOMDR 00000000 00000000 00000000 00000000			
0x0000007C	EBI_ERRR 00000000 00000000 00000000 00000000			

42.2.1 Unlock Register (EBI_UNLOCK)

This register decides the write accessibility of the configuration registers of EBI module. It has to be written with the specific unlock value (0xEB14_10CE) to reset the EBI_LSTSR:LOCKSTATUS bit, which allows write access to the configuration registers. After completion of the configuration the EBI_UNLOCK register should be written with the specific lock (0x10CE_0EB1) value, which sets the EBI_LSTSR:LOCKSTATUS bit and restricts write access to the configuration register.

Unlock Register (EBI_UNLOCK)

Figure 42-3. Unlock Register (EBI_UNLOCK)

EBI_UNLOCK																															
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**
UNLOCK[31]	UNLOCK[30]	UNLOCK[29]	UNLOCK[28]	UNLOCK[27]	UNLOCK[26]	UNLOCK[25]	UNLOCK[24]	UNLOCK[23]	UNLOCK[22]	UNLOCK[21]	UNLOCK[20]	UNLOCK[19]	UNLOCK[18]	UNLOCK[17]	UNLOCK[16]	UNLOCK[15]	UNLOCK[14]	UNLOCK[13]	UNLOCK[12]	UNLOCK[11]	UNLOCK[10]	UNLOCK[9]	UNLOCK[8]	UNLOCK[7]	UNLOCK[6]	UNLOCK[5]	UNLOCK[4]	UNLOCK[3]	UNLOCK[2]	UNLOCK[1]	UNLOCK[0]
R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp	R0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-3. Unlock Register (EBI_UNLOCK) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	UNLOCK	External Bus Interface Lock/Unlock This Unlock Register protects the EBI registers from being modified accidentally by software. The configuration and exception vector registers cannot be written until this register has been written with unlock (0xEB14_10CE) value. To lock the EBI registers again, the software must write the lock (0x10CE_0EB1) value. Illegal access to configuration registers or writing a value other than the lock/unlock value to this register causes a protection error. The list of registers in the EBI which are protected by this register are: ■ EBI_LSTSR ■ EBI_SFMR0~7 ■ EBI_SFACCR0~7 ■ EBI_SFADDCR0~7 ■ EBI_SDMODCR ■ EBI_SDRCR ■ EBI_SDPCR ■ EBI_SDTCR ■ EBI_SDCOMDR ■ EBI_ERRR Reading to this register always returns '0'.

42.2.2 Lock Status Register (EBI_LSTSR)

This register provides the status of EBI lock feature.

Lock Status Register (EBI_LSTSR)

Figure 42-4. Lock Status Register (EBI_LSTSR)

EBI_LSTSR																												
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	LOCKSTATUS
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 42-4. Lock Status Register (EBI_LSTSR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	LOCKSTATUS	External Bus Interface Lock Status This bit provides the locked/unlocked status of the EBI configuration registers. '0': Configuration registers are unlocked for write access '1': Configuration registers are locked for write access By default this bit is '1', writing the correct unlock value to EBI_UNLOCK:UNLOCK resets this bit to '0'. Writing the correct lock value to EBI_UNLOCK:UNLOCK sets this bit to '1'.

42.2.3 SRAM/FLASH Mode Control Register (EBI_SFMR0~7)

This register provides the mode configuration for SRAM/FLASH. EBI_SFMRn:RDY, EBI_SFMRn:PAGE and EBI_SFMRn:NAND should be set mutually exclusive. Each of these 8 registers are identical in function.

Figure 42-5. SRAM/FLASH Mode Control Register (EBI_SFMR0)

Figure 42-6. SRAM/FLASH Mode Control Register (EBI_SFMR0)

EBI_SFMR0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENDIANESS	RDY	PAGE	NAND	WEOFF	RBMON	WDTH[1]	WDTH[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-5. SRAM/FLASH Mode Control Register (EBI_SFMR0) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	ENDIANESS	SRAM/FLASH Endianess Controller These bits provide endianess control for 8 SRAM/FLASH memory locations. '0': SRAM/FLASH memory bank 0 is in little endian mode for access '1': SRAM/FLASH memory bank 0 is in big endian mode for access
[6]	RDY	SRAM/FLASH Ready Mode Set this bit to '1' to have a handshake with a low-speed peripheral that uses the iMRDY signal. At least, four or more cycles (oMCLK) must be inserted between the falling edge of iMRDY and the falling edge of the read/write strobe signal. Set this bit to '0' when a device such as an SRAM memory which is not using the ready signal, is accessed. '0': Ready mode OFF '1': Ready mode ON **Note:** The behaviour of the EBI is undefined if EBI_SFMR0:NAND, EBI_SFMR0:PAGE and RDY are not set mutually exclusive.

Table 42-5. SRAM/FLASH Mode Control Register (EBI_SFMR0) bits

Bit Position	Bit Field Name	Bit Description
[5]	PAGE	SRAM/FLASH NOR FLASH Page Access Mode This bit controls the NOR FLASH page access mode. In the NOR FLASH page access mode, the first address cycle is issued according to the setting of the first read address cycle (EBI_SFACCR0:FRADC). After that, accesses are continuously executed until it reaches 16 bytes boundary according to the setting of the read access cycle (EBI_SFACCR0:RACC). When this mode is selected, set EBI_SFMR0:RBMON to '0' and the read address cycle (EBI_SFACCR0:RADC) to '0'. '0': NOR FLASH page access mode OFF ' 1': NOR FLASH page access mode ON **Note:** The behaviour of EBI is undefined if EBI_SFMR0:NAND, EBI_SFMR0:RDY and PAGE are not set mutually exclusive.
[4]	NAND	SRAM/FLASH NAND FLASH Mode This bit controls the mode to connect NAND FLASH. To enable the access to NAND FLASH, this bit must be set to '1'. In NAND FLASH mode, the corresponding oMCSX is fixed to '0' and the NAND FLASH dedicated pin is used in accesses after this. Reset this bit to '0' when NAND is not used and the oMCSX is fixed to '1', and the NAND FLASH is kept in a power saving state. '0': NAND FLASH mode OFF '1': NAND FLASH mode ON **Note:** The behaviour of EBI is undefined if EBI_SFMR0:PAGE, EBI_SFMR0:RDY and NAND are not set mutually exclusive.
[3]	WEOFF	SRAM/FLASH Write Enable OFF Disables the write enable signal (oMWEX) operation. Saves current consumption by prohibiting unnecessary operation of oMWEX when using the byte mask signal (oMDQM) as the write enable of the device. When disabled, oMWEX is fixed to '1'. '0': Enabled '1': Disabled
[2]	RBMON	SRAM/FLASH Read Byte Mask ON Enables byte mask signal (oMDQM) at read access. Controls unnecessary data outputs from devices on which byte mask is possible and saves power consumed by the total accessing. '0': Disabled '1': Enabled

Table 42-5. SRAM/FLASH Mode Control Register (EBI_SFMR0) bits

Bit Position	Bit Field Name	Bit Description
[1:0]	WDTH	SRAM/FLASH Data Width Data bit width of a connected device is specified. '00': 8 bits '01': 16 bits '10': 32 bits '11': Reserved

42.2.4 SRAM/FLASH Access Configuration Register (EBI_SFACCR0~7)

This register provides the configuration bits to control the access timing for SRAM/FLASH memory. Each of these 8 registers are identical in function.

SRAM/FLASH Access Configuration Register (EBI_SFACCR0)

Figure 42-7. SRAM/FLASH Access Configuration Register (EBI_SFACCR0)

EBI_SFACCR0																															
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**
WIDLC[3]	WIDLC[2]	WIDLC[1]	WIDLC[0]	WWEC[3]	WWEC[2]	WWEC[1]	WWEC[0]	WADC[3]	WADC[2]	WADC[1]	WADC[0]	WACC[3]	WACC[2]	WACC[1]	WACC[0]	RIDLC[3]	RIDLC[2]	RIDLC[1]	RIDLC[0]	FRADC[3]	FRADC[2]	FRADC[1]	FRADC[0]	RADC[3]	RADC[2]	RADC[1]	RADC[0]	RACC[3]	RACC[2]	RACC[1]	RACC[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1

Table 42-6. SRAM/FLASH Access Configuration Register (EBI_SFACCR0) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	WIDLC	SRAM/FLASH Write Idle Cycle These bits set the number of idle cycles after a write access. Configure 0x2 or a greater value when EBI_SFMR0:RDY bit is set to '1'. '0000': 1 cycle '0001': 2 cycles ... '1111': 16 cycles **Note:** When a low-speed device with ready function is used, WIDLC is one cycle less for subsequent accesses of a splitt access.
[27:24]	WWEC	SRAM/FLASH Write Enable Cycle These bits set the number of cycles for a write enable assertion. The setting of these bits also affects the oMDQM (byte mask signal). '0000': 1 cycle '0001': 2 cycles ... '1110': 15 cycles '1111': Reserved

Table 42-6. SRAM/FLASH Access Configuration Register (EBI_SFACCR0) bits

Bit Position	Bit Field Name	Bit Description
[23:20]	WADC	SRAM/FLASH Write Address Setup Cycle These bits set the number of setup cycles of the write address. The address is output for the cycles set in these bits, but the write enable is not asserted during such cycles. '0000': 1 cycle '0001': 2 cycles ... '1110': 15 cycles '1111': Reserved
[19:16]	WACC	SRAM/FLASH Write Access Cycle These bits set the number of cycles necessary for a write access. The address is not changed during the cycles specified in these bits. '0000','0001': Reserved '0010': 3 cycles ... '1111': 16 cycles **Note:** The value must be greater than the sum of the address setup cycle and the write enable cycle.
[15:12]	RIDLC	SRAM/FLASH Read Idle Cycle These bits set the number of idle cycles after the read access. The bits are used to avoid a data collision caused by a write access immediately after the read access. Configure 0xF when a low-speed device with ready function is used. '0000': 1 cycle '0001': 2 cycles ... '1111': 16 cycles **Note:** When a low-speed device with ready function is used RIDLC is one cycle less for subsequent accesses of a splitted access.

Table 42-6. SRAM/FLASH Access Configuration Register (EBI_SFACCR0) bits

Bit Position	Bit Field Name	Bit Description
[11:8]	FRADC	FLASH First Read Address Cycle These bits are used exclusively for the setting of NOR FLASH that supports the page mode access. The bits set the initial latency of the address in the read access of the FLASH. The address is held during the specified cycles only in the first accessing. After that, accesses are performed according to the number of cycles set in EBI_SFACCR0:RACC. In the page mode access, oMCSX and oMOEX are asserted at the same time. When any value other than '0' is set in these bits, configure 0x0 in the EBI_SFACCR0:RADC (read address setup cycle). '0000': 0 cycle '0001': 1 cycles ... '1111': 15 cycles
[7:4]	RADC	SRAM/FLASH Read Address Setup Cycle These bits set the number of setup cycles of the read address. In the read address setup cycles, oMCSX and the address is asserted, but oMOEX is not asserted. When 0x0 is selected, oMOEX and oMCSX are asserted at the same time. The value specified here must be within the number of read access cycles (EBI_SFACCR0:RADC less than EBI_SFACCR0:RACC). Configure 0xA (10) to use a low-speed device with ready function. Set this bit to '0' when the NOR FLASH page access mode is used. '0000': 0 cycle '0001': 1 cycles ... '1111': 15 cycles **Note:** In NAND Flash mode (EBI_SFMRn:NAND='1') configure RADC > 0, if the size of an access to the NAND Flash memory range is larger than the width of the external bus (EBI_SFMRn:WDTH).
[3:0]	RACC	SRAM/FLASH Read Access Cycle These bits set the necessary cycles for read access. The address is not changed during the cycles specified in these bits, and data is fetched in the final cycle. Configure 0xF to use a low-speed device with ready function. '0000': 1 cycle '0001': 2 cycles ... '1111': 16 cycles

42.2.5 SRAM/FLASH Address Control Register (EBI_SFADDCR0~7)

This register provides the address and address mask to select a specific memory bank. Each of these 8 registers are identical in function.

SRAM/FLASH Address Control Register (EBI_SFADDCR0)

Figure 42-8. SRAM/FLASH Address Control Register (EBI_SFADDCR0)

EBI_SFADDCR0																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	MASK[7]	MASK[6]	MASK[5]	MASK[4]	MASK[3]	MASK[2]	MASK[1]	MASK[0]	read0	read0	read0	read0	read0	read0	read0	read0	ADDR[7]	ADDR[6]	ADDR[5]	ADDR[4]	ADDR[3]	ADDR[2]	ADDR[1]	ADDR[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
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	0	0	0	0	0

Table 42-7. Initial values for registers EBI_SFADDCR1 to EBI_SFADDCR7

Address Control Register	Initial Value
EBI_SFADDCR1:	0x000F0010
EBI_SFADDCR2:	0x000F0020
EBI_SFADDCR3:	0x000F0030
EBI_SFADDCR4:	0x000F0040
EBI_SFADDCR5:	0x000F0050
EBI_SFADDCR6:	0x000F0060
EBI_SFADDCR7:	0x000F0070

Table 42-8. SRAM/FLASH Address Control Register (EBI_SFADDCR0) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:16]	MASK	SRAM/FLASH Mask Address These bits set the mask value of the value set in EBI_SFADDCR0:ADDR. The EBI masks the EBI_SFADDCR0:ADDR and the internal bus address (masked when '1' is set) according to the specified mask and compares them. When they match, it accesses the oMCSX signal.
[15:8]	read0	-

Table 42-8. SRAM/FLASH Address Control Register (EBI_SFADDCR0) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	ADDR	SRAM/FLASH Address register These bits specify addresses to set the corresponding oMCSX region. The addresses are those in the fixed area of the 256 MB assigned to the SRAM/FLASH IF. Value equivalent to part of the addresses [27:20] must be defined. **Note:** Each address field must not overlap

MASK bit Example:

When they match, it accesses the oMCSX signal. [22:16] mask addresses [26:20] respectively. Setting for 64 MB and 128 MB is not permitted because the bit width of oMAD (external pin) is 25 bits.

Example:

EBI_SFADDCR:ADDR = 00001000 (b)

EBI_SFADDCR:MASK = 00000011 (b)

When to be selected

Internal bus address (address of the external I/F) oMAD = 0x10900000

Masking

EBI_SFADDCR:ADDR & (!MASK) = 00001000 (b)

EBI_SFADDCR:AD [27:20] & (!MASK) = 00001000 (b).. Match. The device is selected.

When not to be selected

Internal bus address (address of the external I/F) oMAD = 0x10C00000

Masking

EBI_SFADDCR:ADDR & (!MASK) = 00001000 (b)

EBI_SFADDCR:AD [27:20] & (!MASK) = 00001100 (b).. No match. The device is not selected.

The masking selects the size of the region. In the example, 0x10800000~0x10BFFFFF (4 MB) are selected. The bits for which '1' is specified in masking are lost in the mask processing. The bits are invalidated even if they have been set in EBI_SFADDCR:ADDR. If the Least Significant Bit (LSB) in the example is '1' (EBI_SFADDCR:ADDR=0x09), the same address field is selected since it is invalidated in the masking. The relationship between mask settings and the sizes of address field is.

0000000 (b) -> 1 MB 0001111 (b) -> 16 MB

0000001 (b) -> 2 MB 0011111 (b) -> 32 MB

0000011 (b) -> 4 MB 0111111 (b) -> 64 MB (Reserved)

0000111 (b) -> 8 MB 1111111 (b) -> 128 MB (Reserved)

Note: Refer to the device-specific datasheet for the actual number of external pins of oMAD.

42.2.6 SDRAM Mode Control Register (EBI_SDMODCR)

This register provides the mode configuration for SRAM/FLASH of EBI.

SDRAM Mode Control Register (EBI_SDMODCR)

Figure 42-9. SDRAM Mode Control Register (EBI_SDMODCR)

EBI_SDMODCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	MCLKOFF	BASEL[3]	BASEL[2]	BASEL[1]	BASEL[0]	RASEL[3]	RASEL[2]	RASEL[1]	RASEL[0]	read0	read0	CASEL[1]	CASEL[0]	read0	ROFF	PDON	SDON
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0

Table 42-9. SDRAM Mode Control Register (EBI_SDMODCR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	MCLKOFF	Memory Clock OFF This bit stops the clock output (oMCLK) of the chip (the power down mode, on the other hand, is set for the clock gate of the SDRAM). The setting of this bit has no relation to the status of EBI_SDMODCR:SDON. When this bit is set to '1' with the EBI_SDMODCR:SDON = '1', clocks are not provided to the SDRAM, and accessing is disabled. When clock provision is resumed with the bit changed from '1' to '0', the delay between oHCLK and oMCLK is generated. This bit should not be changed at the same time as the EBI_SDMODCR:SDON bit is changed to '1'. '0': Clock output ON '1': Clock output OFF

Table 42-9. SDRAM Mode Control Register (EBI_SDMODCR) bits

Bit Position	Bit Field Name	Bit Description
[15:12]	BASEL	Bank Address Select Address bits on the internal bus outputs as bank addresses are selected. '0000': A[15:14] = Internal address [20:19] '0001': A[15:14] = Internal address [21:20] '0010': A[15:14] = Internal address [22:21] '0011': A[15:14] = Internal address [23:22] '0100': A[15:14] = Internal address [24:23] '0101': A[15:14] = Internal address [25:24] '0110': A[15:14] = Internal address [26:25] Others: Reserved
[11:8]	RASEL	Row Address Select Address bits on the internal bus outputs as row addresses are selected. '0000': A[13:0] = Internal address [19:6] '0001': A[13:0] = Internal address [20:7] '0010': A[13:0] = Internal address [21:8] '0011': A[13:0] = Internal address [22:9] '0100': A[13:0] = Internal address [23:10] '0101': A[13:0] = Internal address [24:11] '0110': A[13:0] = Internal address [25:12] Others: Reserved
[7:6]	read0	-
[5:4]	CASEL	Column Address Select Address bits on the internal bus output as column addresses are selected. The setting is also used for the setting of the SDRAM bus width. '00': A[9:0] = Internal address [10:1], 16-bit width '01': A[9:0] = Internal address [11:2], 32-bit width '10': Reserved '11': Reserved
[3]	read0	-

Table 42-9. SDRAM Mode Control Register (EBI_SDMODCR) bits

Bit Position	Bit Field Name	Bit Description
[2]	ROFF	Refresh OFF While accessing the SDRAM command register this bit is used to suspend refresh, etc. Although the refresh counter continues to be active, no refresh is executed even in refresh timing. When ROFF is '0', SDRAM is refreshed if the refresh time has already passed with no refresh. SDRAM is refreshed only once if two or more refresh timings have passed. When the data needs to be held, make such settings that the suspension will not go beyond the refresh timing, or explicitly execute refresh by accessing the SDRAM command register. '0': Refresh is active '1': Refresh is suspended
[1]	PDON	Power Down mode ON Power-down mode is used. With this mode ON, the SDRAM enters the power-down mode if no memory access occurs during the specified period, saving power (oMCKE = '0'). During refresh, SDRAM is reset and refreshed. After that the mode again enters the power-down mode if no access is made during the specified period. The power-down count is reset one cycle after the internal bus access because the latency from internal bus to external SDRAM is one clock cycle. '0': OFF '1': ON The mode does not a issue self-refresh command to the SDRAM **Note:** Self refresh does not work if PD3 is powered down in Power saving state (PSS). It will work if the PD3 clocks are simply disabled

Table 42-9. SDRAM Mode Control Register (EBI_SDMODCR) bits

Bit Position	Bit Field Name	Bit Description
[0]	SDON	SDRAM ON Control This bit enables access to SDRAM. The bit issues a power-on sequence to the SDRAM, and automatically sets the mode registers, etc. When the bit is set to '0' during operation, all accesses to the SDRAM is stopped after completion of the accesses (during refresh, after the refresh ((TREFC + 1)*(NREF + 1)). When EBI_SDMODCR:BASEL, EBI_SDMODCR:RASEL, and EBI_SDMODCR:CASEL are changed until the completion of the accesses, incorrect operations may happen. Be careful when setting it OFF during operation. When EBI_SDMODCR:PDON = '1', oMCKE = '0'. In this case, data is not held because no refresh is executed. An error is returned for an access. When data is written to the SDRAM command register at OFF, the External Bus Interface automatically sets this bit ON(= '1') without issuing a power-on sequence since it expects that the user starts up the SDRAM (set EBI_SDMODCR:ROFF to '1' (refresh OFF) in this case). '0': OFF '1': ON Configure SDRAM regsiters for the required operation before setting this bit to '1'.

Figure 42-10. SDRAM automatic set mode register sequence

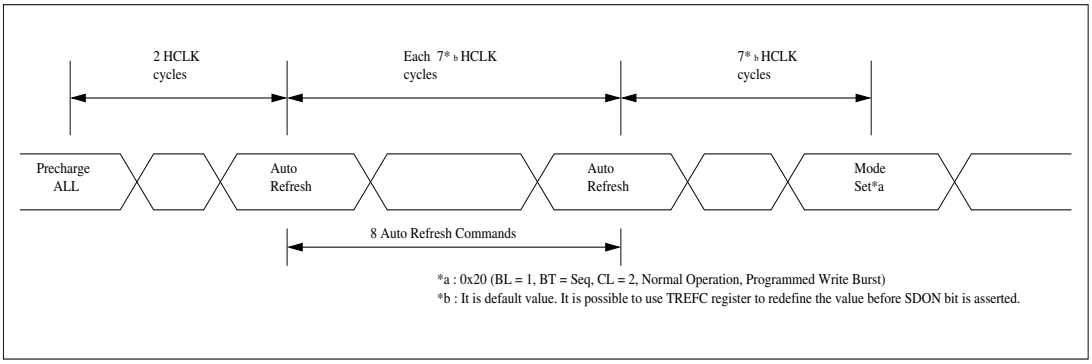
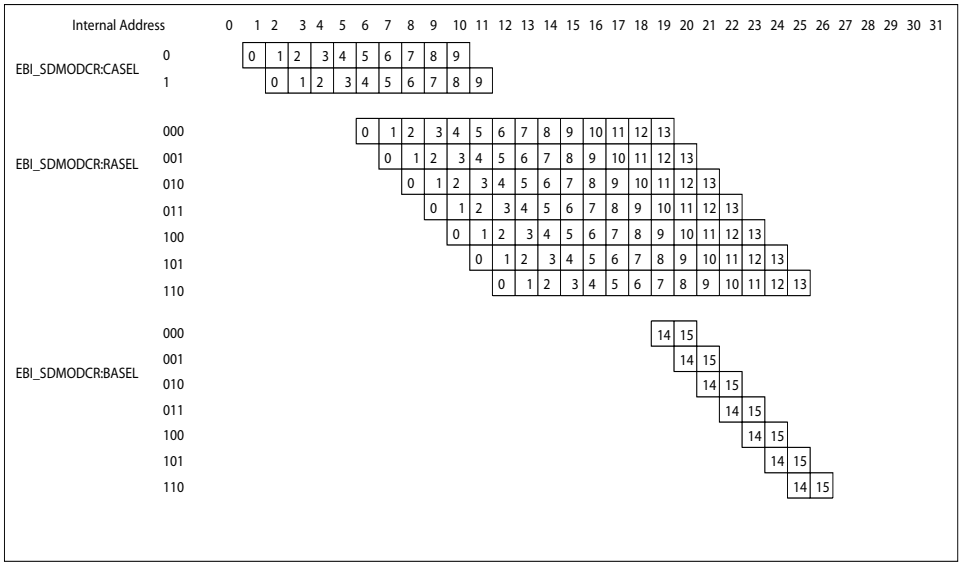


Figure 42-11. Figure showing correspondence between EBI_SDMODCR:CASEL, EBI_SDMODCR:RASEL and EBI_SDMODCR:BASEL



42.2.7 SDRAM Refresh Control Register (EBI_SDRCR)

This register provides the configuration bits to configure the refresh timings.

SDRAM Refresh Control Register (EBI_SDRCR)

Figure 42-12. SDRAM Refresh Control Register (EBI_SDRCR)

EBI_SDRCR																															
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**
read0	read0	read0	read0	read0	read0	read0	PREF	NREF[7]	NREF[6]	NREF[5]	NREF[4]	NREF[3]	NREF[2]	NREF[1]	NREF[0]	REFC[15]	REFC[14]	REFC[13]	REFC[12]	REFC[11]	REFC[10]	REFC[9]	REFC[8]	REFC[7]	REFC[6]	REFC[5]	REFC[4]	REFC[3]	REFC[2]	REFC[1]	REFC[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0

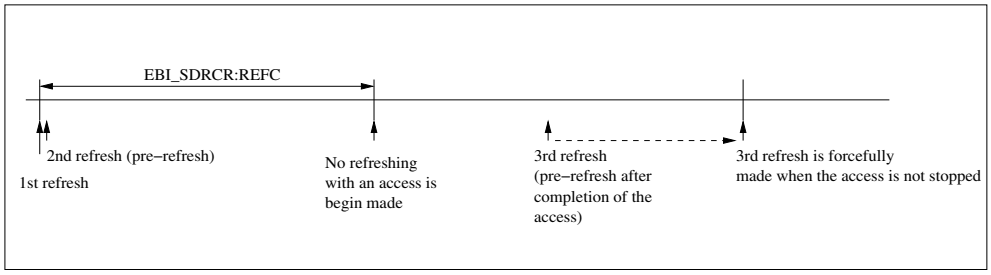
Table 42-10. SDRAM Refresh Control Register (EBI_SDRCR) bits

Bit Position	Bit Field Name	Bit Description
[31:25]	read0	-
[24]	PREF	Pre-refresh ON Sets the mode to execute a refresh earlier than a specified timing (pre- refresh). Refresh at burst accessing temporarily lowers transfer efficiency, consuming more power because of precharging. To prevent this, a pre-refresh is made in an earlier timing when there is no access after the last refresh has been executed. In the next refresh timing, no refresh will be made because the one expected to happen in this timing has already been executed. When a burst access happens in that succeeding refresh timing, the accessing can continue. If the burst access has followed the last refresh and no pre-refresh occurs, the burst access will be suspended in the next refresh timing, forcefully executing the refresh. '0': Pre-refresh disabled '1': Pre-refresh enabled

Table 42-10. SDRAM Refresh Control Register (EBI_SDRCR) bits

Bit Position	Bit Field Name	Bit Description
[23:16]	NREF	Number of Refresh These bits set the number of times a refreshed is issued in one refresh timing. No access is permitted during refresh. Intensive refresh in a certain period is allowed. Calculate an appropriate value of the EBI_SDRCR:REFC (Refresh Counts) for the NREF using the following equation and specify it to REFC. REFC = (Refresh counts in one refresh timing) * (NREF + 1) 0x0: 1 time 0x1: 2 times ... 0xFF:256 times
[15:0]	REFC	Refresh Count This register sets refresh intervals. After specified cycles (oMCLK) have passed, a refresh is executed. When data is written during an access, the current access is suspended and the first refresh is made. The minimum value of the intervals must not be smaller than the cycles required for one refresh (TREFC + 1). If a smaller value is specified, it may cause a hang up. 0x0:Reserved, ... 0x9: Reserved 0xA: 11 cycles 0xB: 12 cycles ... 0xFFFF:65536 cycles

Figure 42-13. Prerefresh timing example



42.2.8 SDRAM Power Control Register (EBI_SDPCR)

This register provides the configuration bits to control the power of SDRAM memory.

SDRAM Power Control Register (EBI_SDPCR)

Figure 42-14. SDRAM Power Control Register (EBI_SDPCR)

EBI_SDPCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	PDC[15]	PDC[14]	PDC[13]	PDC[12]	PDC[11]	PDC[10]	PDC[9]	PDC[8]	PDC[7]	PDC[6]	PDC[5]	PDC[4]	PDC[3]	PDC[2]	PDC[1]	PDC[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-11. SDRAM Power Control Register (EBI_SDPCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	PDC	Power Down Count These bits set the counts for the mode to transit to the power down mode. When an access to the SDRAM does not occur during the cycles specified here (oMCLK), the mode transits to the power down. 0x0:0 cycle 0x1:2 cycles ... 0xFFFF:65535 cycles

42.2.9 SDRAM Timing Control Register (EBI_SDTCR)

This register provides the configuration bits to control various timing modes related to SDRAM External Bus Interface.

SDRAM Timing Control Register (EBI_SDTCR)

Figure 42-15. SDRAM Timing Control Register (EBI_SDTCR)

EBI_SDTCR																															
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**
read0	read0	read0	read0	read0	read0	TDPL[1]	TDPL[0]	TREFC[3]	TREFC[2]	TREFC[1]	TREFC[0]	TRAS[3]	TRAS[2]	TRAS[1]	TRAS[0]	TRCD[3]	TRCD[2]	TRCD[1]	TRCD[0]	TRP[3]	TRP[2]	TRP[1]	TRP[0]	TRC[3]	TRC[2]	TRC[1]	TRC[0]	read0	read0	CL[1]	CL[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	RpWp	RpWp
0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1

Table 42-12. SDRAM Timing Control Register (EBI_SDTCR) bits

Bit Position	Bit Field Name	Bit Description
[31:26]	read0	-
[25:24]	TDPL	Data in to Precharge Lead Time These bits set the latency from write to precharge. '00': 1 cycle '01': 2 cycles '10': 3 cycles '11': 4 cycles
[23:20]	TREFC	Refresh Cycle time These bits set the latency of the command following a refresh. '0000': 1 cycle '0001': 2 cycles '0010': 3 cycles '0011': 4 cycles '0100': 5 cycles '0101': 6 cycles '0110': 7 cycles '0111': 8 cycles Others: Reserved

Table 42-12. SDRAM Timing Control Register (EBI_SDTCR) bits

Bit Position	Bit Field Name	Bit Description
[19:16]	TRAS	RAS Active Time These bits set the minimum active time of ROW. '0000': 1 cycle '0001': 2 cycles '0010': 3 cycles '0011': 4 cycles '0100': 5 cycles '0101': 6 cycles '0110': 7 cycles '0111': 8 cycles Others: Reserved
[15:12]	TRCD	RAS CAS Delay These bits set the latency from RAS to CAS. '0000': 1 cycle '0001': 2 cycles Others: Reserved
[11:8]	TRP	RAS Precharge Time These bits set the time of precharging. '0000': 1 cycle '0001': 2 cycles '0010': 3 cycles '0011': 4 cycles Others: Reserved
[7:4]	TRC	RAS Cycle Time These bits set the latency from RAS to RAS. '0000': 1 cycle '0001': 2 cycles '0010': 3 cycles '0011': 4 cycles '0100': 5 cycles '0101': 6 cycles '0110': 7 cycles '0111': 8 cycles Others: Reserved
[3:2]	read0	-
[1:0]	CL	CAS Latency These bits set CAS latency. '00': 1 cycle '01': 2 cycles '10': 3 cycles '11': Reserved

42.2.10 SDRAM Command Register (EBI_SDCOMDR)

This register provides the command control of SDRAM.

SDRAM Command Register (EBI_SDCOMDR)

Figure 42-16. SDRAM Command Register (EBI_SDCOMDR)

EBI_SDCOMDR																															
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**
PEND	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SDCKE	SDCS	SDRAS	SDCAS	SDWE	SDAD[15]	SDAD[14]	SDAD[13]	SDAD[12]	SDAD[11]	SDAD[10]	SDAD[9]	SDAD[8]	SDAD[7]	SDAD[6]	SDAD[5]	SDAD[4]	SDAD[3]	SDAD[2]	SDAD[1]	SDAD[0]
Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-13. SDRAM Command Register (EBI_SDCOMDR) bits

Bit Position	Bit Field Name	Bit Description
[31]	PEND	Pending Access Wait This bit is asserted immediately after a data write to this register has been detected. This is because the command to the SDRAM (the written value) cannot be executed soon due to access to other devices, etc. Ensure that this bit is deasserted before writing data to this register. Operations cannot be guaranteed if writing is made during assertion. '0': Not pending '1': Pending
[30:21]	read0	-
[20]	SDCKE	SDRAM CKE Detects data write to the register and outputs a specified value to oMCKE pin.
[19]	SDCS	SDRAM Chip Select Detects data write to the register and outputs a specified value to oMCSX.
[18]	SDRAS	SDRAM RAS Detects data write to the register and outputs a specified value to oMRASX.

Table 42-13. SDRAM Command Register (EBI_SDCOMDR) bits

Bit Position	Bit Field Name	Bit Description
[17]	SDCAS	SDRAM CAS Detects data write to the register and outputs a specified value to oMCASX.
[16]	SDWE	SDRAM Write Enable Detects data write to the register and outputs a specified value to oMWEX.
[15:0]	SDAD	SDRAM Address These bits detect data write to the register and outputs a specified value to A[15:0].

Note:

When data write to this register has been detected, the written values specified to this register (except bit EBI_SDCOMDR:SDCKE) will be output for one cycle. Values of D[31:0] are held during this period. Usually, EBI_SDCOMDR:PEND is not required in the EBI since SDRAM power on sequence is automatically executed. It is used for setting (of extend mode, etc.) during operation. When EBI_SDCOMDR:SDCKE = ’0’, the mode enters the power down mode. The status of the EBI_SDMODCR:PDON bit of the SDRAM mode register is not involved with the operation. The mode is reset by an access to the SDRAM, refresh or write to the register. To stop the refresh, set the EBI_SDMODCR:ROFF to ’1’ (refresh OFF).

42.2.11 Error Register (EBI_ERRR)

This register provides the Error status for SRAM/FLASH or SDRAM access.

Error Register (EBI_ERRR)

Figure 42-17. Error Register (EBI_ERRR)

EBI_ERRR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SDER	SFER
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp1	RpWp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42-14. Error Register (EBI_ERRR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:2]	read0	-
[1]	SDER	SDRAM Error Indicates an access has been made to SDRAM with the SDRAM mode register set to '0'. In this case the External Bus Interface returns an error response to the internal bus. '0': No error '1': Error Writing '1' to this bit clears the SDRAM error flag. Writing '0' to this register does not have effect.
[0]	SFER	SRAM/FLASH Error Indicates an access has been made to the region that is not mapped during an access to the SRAM/FLASH. In this case the EBI returns an error response to the internal bus. '0': No error '1': Error Writing '1' to this bit clears the SRAM/FLASH error flag. Writing '0' to this register does not have any effect.

42.3 Operation of the External Bus Interface

The EBI can control SRAM, FLASH and SDRAM. Internally the EBI consists of the SRAM/FLASH controller and the SDRAM controller. This chapter describes the operation of the EBI in detail.

42.3.1 SRAM/FLASH controller

The SRAM/FLASH controller has an address space of 256 MB. Each address can be set optionally (with the bit width of the external output addresses taken into account, the maximum size of an actual address is 32 MB). A different timing and endianess can be specified for each chip select. NAND FLASH and NOR FLASH can be connected. NOR FLASH is accessed by the usual SRAM accessing. Dedicated pins are provided for NAND FLASH. With this, access to the device during a NAND access, sharing a data pin is enabled (see Figure 42-27). When the access is made using a bit width greater than the targeted one, the access is converted into a continuous access where only addresses are changed with the oMCSX is kept to ‘0’. For example, when a 32-bit read access is performed to an 8-bit width device from the internal bus and the device is not using the READY signal to control the data access, the address transits 0->1->2->3 while oMCSX being kept ‘0’, and data are sequentially fetched from D[7:0] according to the transit timings (see 42.5 Access waveform examples for the External Bus Interface). Section n describes the senario when the READY signal is used to control data access.Low-speed device interface function. Data read from the device to the internal bus depend on the configured endianess. When access is made using a bit width narrower than the targeted one (e.g. byte access to a 16-bit width target), byte access execution will be controlled by the oMDQM (byte mask) signal if it is in write operation (The SRAM/FLASH controller outputs only the necessary data). Concerning a device without an input mask, the oMDQM signal can be used as a write enable. During a read access, if the target device has an output mask signal, unnecessary outputs from the device can be avoided by letting the device output only the data of necessary bytes using the oMDQM control. As a result, power dissipation is reduced.

Low-speed device interface function

The (EBI) can be connected to the peripheral that has a general purpose low-speed interface by connecting to the iMRDY pin of the EBI to the peripheral ready (READY) signal. The iMRDY pins should be in ‘wait’ at ‘0’ and in ‘ready’ at ‘1’. Setting the iMRDY to ‘0’ must be synchronous with the falling edge of the oMCSX signal. In an access exceeding the external data bus width (e.g. word (32 bits) access to an 8-bit device), accesses will be generated in succession ‘read’ -> ‘read’, ‘write’ -> ‘write’ until it covers all the excessive bits. In this case, oMCSX will be high between the two continous reads or writes.The high time for write is defined by the EBI_SFACCRn:WIDLC and for the read it is defined by EBI_SFACCRn:RIDLC. For devices not using the READY function, such as SRAM memory, etc, the EBI_SFMRn:RDY bit of the corresponding chip select must be OFF. If the iMRDY signal is at ‘1’ from the beginning of the access, the access will be executed in the same manner as a usual SRAM access. Operations cannot be guaranteed in case the iMRDY falls (or becomes high pulse) in the course of the access cycle.

Note: The function is not applicable to RDY/BUSY signal of FLASH memory.

42.3.2 SDRAM controller

The SDRAM controller has an address space of 128 MB. It accesses according to the correspondence among the internal bus, row column and BA specified by the user. This enables connections to SDRAMs of various configurations. Either 16 bits or 32 bits can be selected for the bus width. Using the full page of the SDRAM makes burst access to the limit of the 1 KB address without a wait possible (in 32-bit SDRAM, refresh and page-miss are excluded). Random accesses are possible almost without a change in latency. The SDRAM controller supports power down mode: when there is no access made to the cycles (0~65535) defined by the user, the SDRAM enters the power-down mode. As this mode is automatically reset, access can be resumed without the user making any setting (usually, one cycle is required for resetting). In one refresh timing, 1 to 256 refreshes can be made successively depending on the setting. Prerefresh is possible in order to avoid a refresh during a burst access. When SDRAM is not used, all accesses including refreshes to the SDRAM can be cut if the registers are so set. With this, more power can be saved. Any access to the SDRAM is ignored on the internal bus, errors being returned to the master (deadlock will not happen). However, datahold is not ensured in this case.

42.3.3 NAND FLASH access

Access to the NAND FLASH requires the different process from the usual SRAM access. The NAND FLASH has (512+16) bytes of internal registers (8-bit/16-bit NANDs have (1024+32) bytes). Basic data accesses are performed in these units. During a read cycle, several μs to tens of μs are needed for loading data to internal registers, in data write to memory cells, hundreds of μs are required and in block deletion, several ms. The chip select signal must be kept '0' in data load to the internal registers. Therefore, the read/write signals to the NAND FLASH cannot be controlled. The EBI, which has a discrete enable signal for NAND, executes accesses with the chip select signal connected to the NAND FLASH held at '0'. (It can be reset to '1'). With this, accessing other chips that share the data line becomes possible. From the user point of view, accesses made to the NAND FLASH are I/O accesses:

1. Write access to +0x2000 is converted to address issue (oMNALE is asserted) to the NAND FLASH.
2. Write access to +0x1000 is converted to command issue (oMNCLE is asserted) to the NAND FLASH.
3. Write/read access to +0x0000 is converted to data access (oMNALE and oMNCLE are not asserted) to the NAND FLASH: based on the base addresses in the region set to NAND mode.

Settings of access timings are the same as those used in SRAM access. oMNCLE is output in the same timing as the address output of the access. oMNALE is kept asserted after the address issue until the write accessing to +0x3000 or any other write accessing (data or command) other than the address issue. This is because the NAND FLASH is incapable of deasserting the oMNALE in multiple write accesses made for the address issue. In the access to +0x3000, only assertion of oMNALE is generated and no access is made. The following figures show the steps of the NAND FLASH accesses.

Figure 42-18. NAND read access

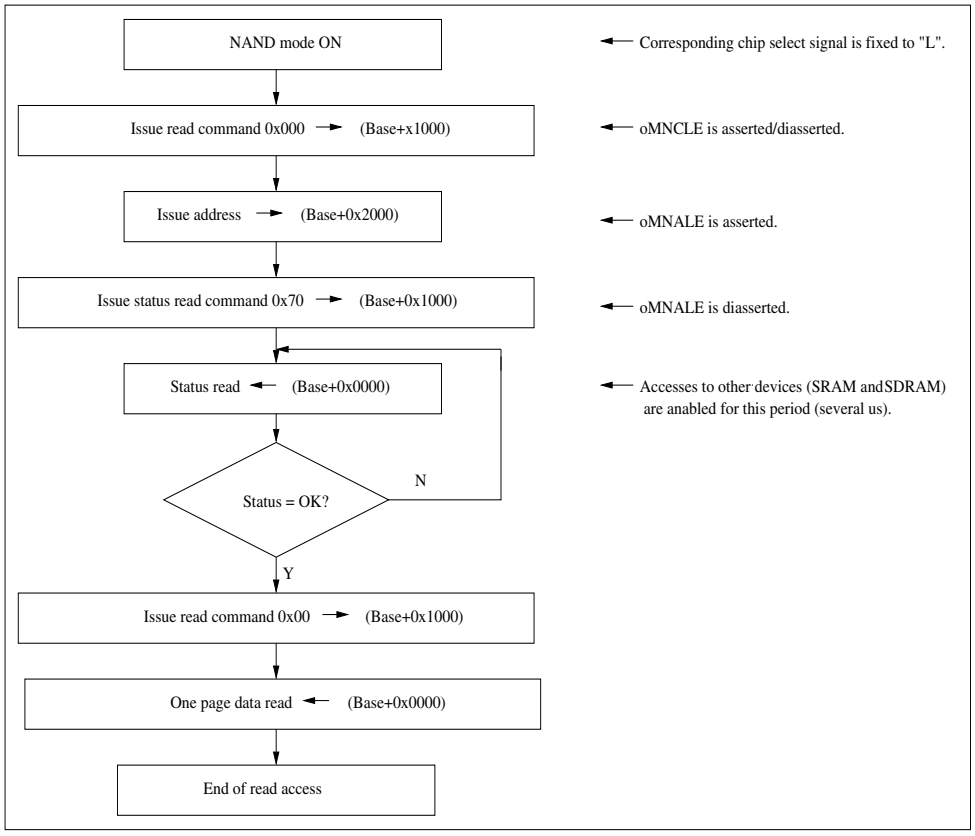


Figure 42-19. NAND FLASH write access (auto program)

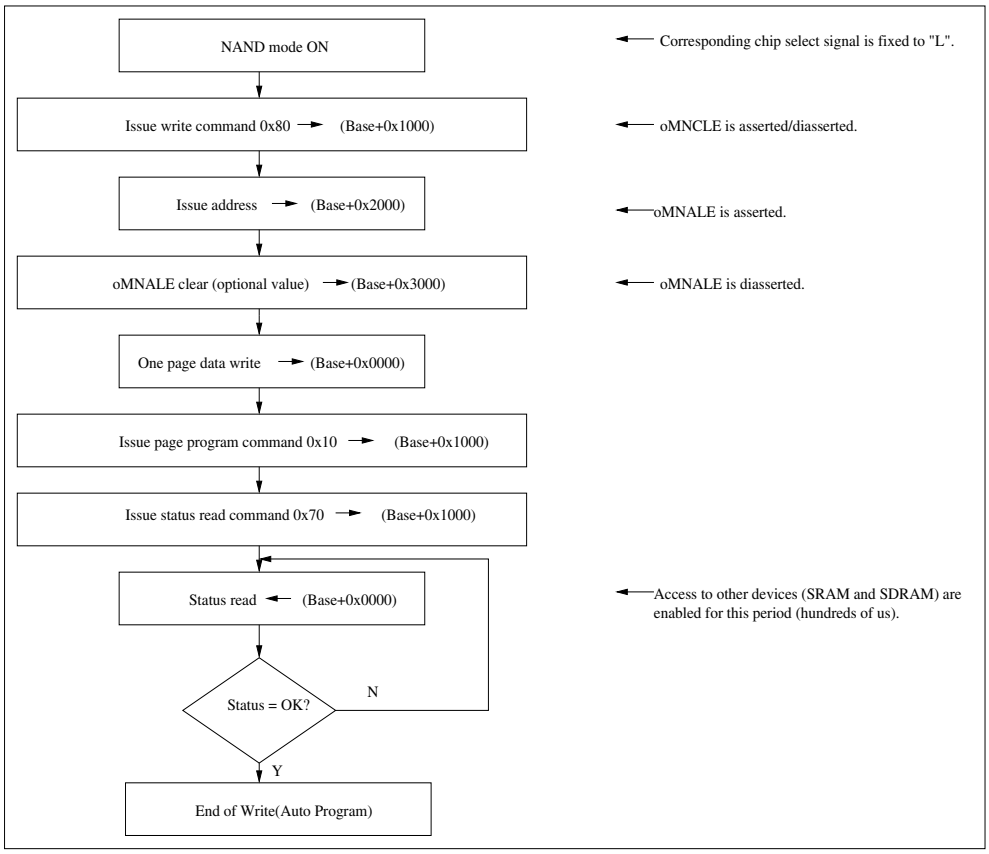
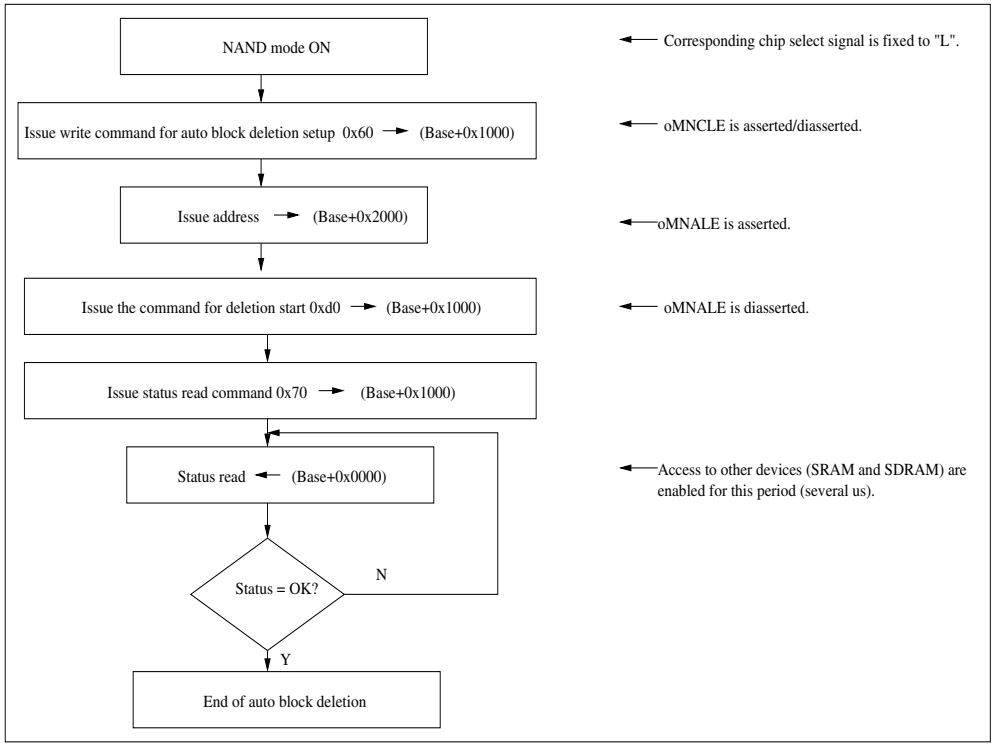


Figure 42-20. NAND FLASH auto block deletion access



As shown in the charts, accesses to other memory devices are enabled even in phases where the NAND FLASH accesses have not been completed. Since DMA can also be used for data read and data write, the processor is capable of making accesses to the NAND with minimum operation.

Note: For details on the interaction of DMA and EBI refer to 42. External Bus Interface.

42.3.4 Endianess and enabled byte lanes

For the EBI, each SRAM/FLASH bank supports bi-endian. The table shows the correspondence between the endianess and byte lanes.

Table 42-15. Endianess access table

Endian	Access size	Target width	Internal bus address	Enabled byte lane	Corresponding AHB data	MDQM [3:0]	MAD [1:0]
Little	Word	8-bit	00	M*Datb[7:0]	H*DATA[7:0]	1110	00
					H*DATA[15:8]		01
					H*DATA[23:16]		10
					H*DATA[31:24]		11
		16-bit	00	M*Datb[15:0]	H*DATA[15:0]	1100	00
				M*Datb[15:0]	H*DATA[31:16]		10
		32-bit	00	M*Datb[31:0]	H*DATA[31:0]	0000	00
	Half word	8-bit	00	M*Datb[7:0]	H*DATA[7:0]	1110	00
					H*DATA[15:8]		01
			10	M*Datb[7:0]	H*DATA[23:16]	1110	10
					H*DATA[23:16]		11
		16-bit	00	M*Datb[15:0]	H*DATA[15:0]	1100	00
			10	M*Datb[15:0]	H*DATA[31:16]	1100	10
		32-bit	00	M*Datb[15:0]	H*DATA[15:0]	1100	00
			10	M*Datb[31:16]	H*DATA[31:16]	0011	00
	Byte	8-bit	00	M*Datb[7:0]	H*DATA[7:0]	1110	00
			01	M*Datb[7:0]	H*DATA[15:8]	1110	01
			10	M*Datb[7:0]	H*DATA[23:16]	1110	10
			11	M*Datb[7:0]	H*DATA[31:24]	1110	11
		16-bit	00	M*Datb[7:0]	H*DATA[7:0]	1110	00
			01	M*Datb[15:8]	H*DATA[15:8]	1101	00
			10	M*Datb[7:0]	H*DATA[23:16]	1110	10
			11	M*Datb[15:8]	H*DATA[31:24]	1101	10
		32-bit	00	M*Datb[7:0]	H*DATA[7:0]	1110	00
			01	M*Datb[15:8]	H*DATA[15:8]	1101	00
			10	M*Datb[23:16]	H*DATA[23:16]	1011	00
			11	M*Datb[31:24]	H*DATA[31:24]	0111	00

Table 42-15. Endianess access table

Endian	Access size	Target width	Internal bus address	Enabled byte lane	Corresponding AHB data	MDQM [3:0]	MAD [1:0]
Big	Word	8-bit	00	M*Datb[7:0]	H*DATA[31:24]	1110	00
					H*DATA[23:16]		01
					H*DATA[15:8]		10
					H*DATA[7:0]		11
		16-bit	00	M*Datb[15:0]	H*DATA[31:16]	1100	00
				M*Datb[15:0]	H*DATA[15:0]		10
		32-bit	00	M*Datb[31:0]	H*DATA[31:0]	0000	00
	Half word	8-bit	00	M*Datb[7:0]	H*DATA[15:8]	1110	00
					H*DATA[7:0]		01
			10	M*Datb[7:0]	H*DATA[31:24]	1110	10
					H*DATA[23:16]		11
		16-bit	00	M*Datb[15:0]	H*DATA[15:0]	1100	00
			10	M*Datb[15:0]	H*DATA[31:16]	1100	10
		32-bit	00	M*Datb[31:16]	H*DATA[15:0]	0011	00
			10	M*Datb[15:0]	H*DATA[31:16]	1100	00
	Byte	8-bit	00	M*Datb[7:0]	H*DATA[7:0]	1110	00
			01	M*Datb[7:0]	H*DATA[15:8]	1110	01
			10	M*Datb[7:0]	H*DATA[23:16]	1110	10
			11	M*Datb[7:0]	H*DATA[31:24]	1110	11
		16-bit	00	M*Datb[15:8]	H*DATA[7:0]	1101	00
			01	M*Datb[7:0]	H*DATA[15:8]	1110	00
			10	M*Datb[15:8]	H*DATA[23:16]	1101	10
			11	M*Datb[7:0]	H*DATA[31:24]	1110	10
		32-bit	00	M*Datb[31:24]	H*DATA[7:0]	0111	00
			01	M*Datb[23:16]	H*DATA[15:8]	1011	00
			10	M*Datb[15:8]	H*DATA[23:16]	1101	00
			11	M*Datb[7:0]	H*DATA[31:24]	1110	00

M*Datb: Read or write data on the external bus.

H*DATA: Read or write data on the internal bus.

Example:

When a byte access is executed to a 32-bit width target, the enabled byte lane varies according to the order of the internal bus addresses shown in the table. Since the data width of the internal bus (32 bits) and the width of the target agree, the data in the internal bus is output to the corresponding byte lane. When a byte access is made to a 16-bit width target, only 2 bytes become enabled according to the endian as shown in the table. Data in the internal bus are distributed to 16 bits according to the endian, and provides the required input and output.

Note: Using external SRAM on the EBI Interface

It is recommended to set the EBI memory region to “Device” or “Strongly Ordered” (by setting the memory range) using the MPU of the Cortex R4 CPU to ensure that memory accesses occur in the same order and have the sizes as the program

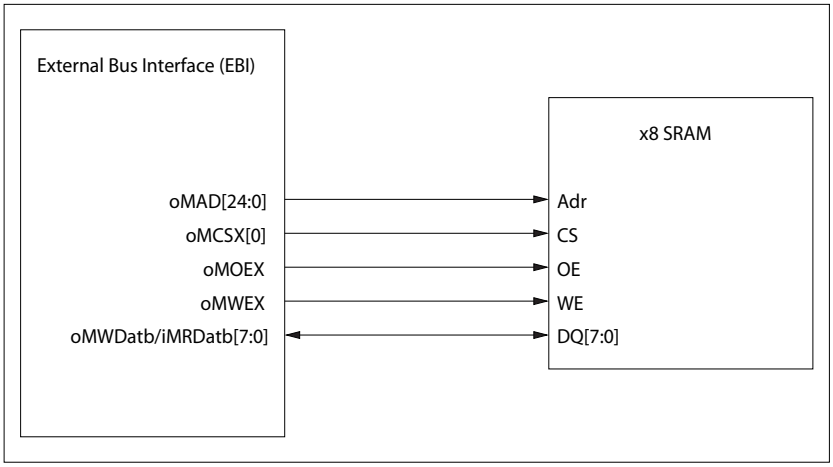
code imply. However, if the memory region needs to be set to “Normal” with “Cachable WBWA” (WBWA = Write Back, Write Allocate), it is important to remember that this setting only works in “little endian” mode on the EBI.

42.4 External memory connection to the External Bus Interface

Examples showing various external memory connection to the EBI are given in this section.

Memory Interface names correspond to the functional pin names, which are similar to the given one.

Figure 42-21. 8-bit SRAM with External Bus Interface



Note:

Inputs/outputs in the above figure which end with an “X”, for example “oMOEX”, indicate active low inputs/outputs

Figure 42-22. Two 8-bit SRAM with External Bus Interface

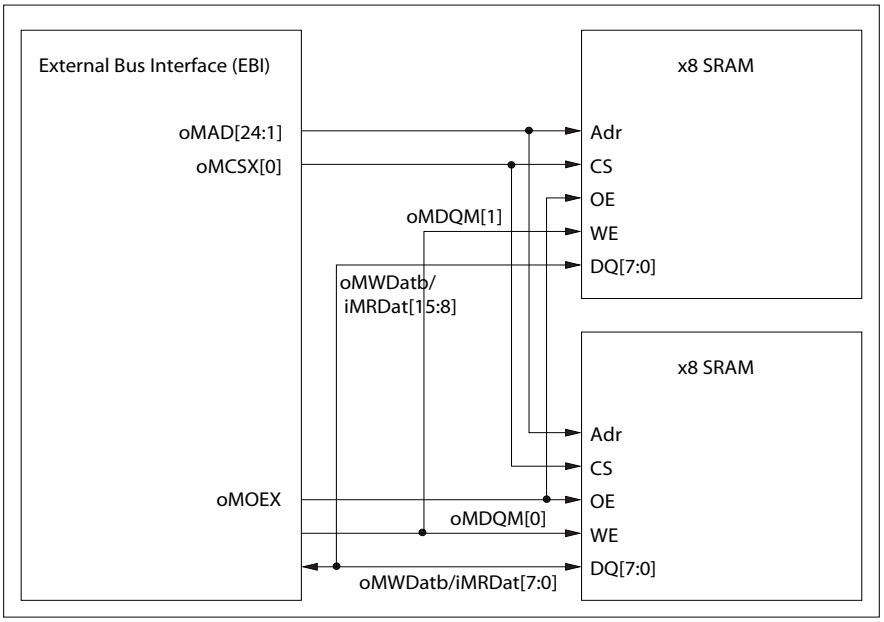
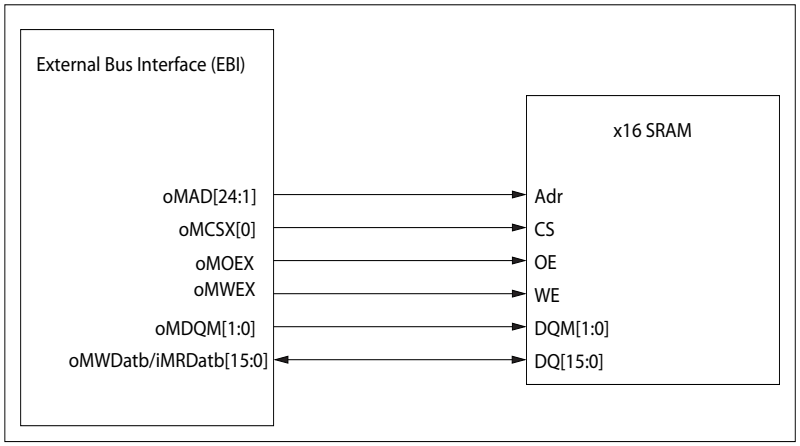
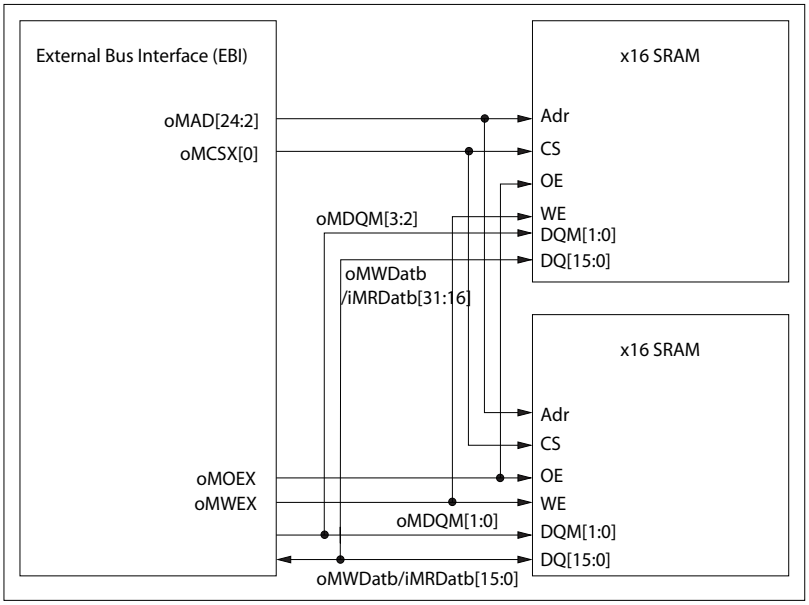


Figure 42-23. 16-bit SRAM with External Bus Interface



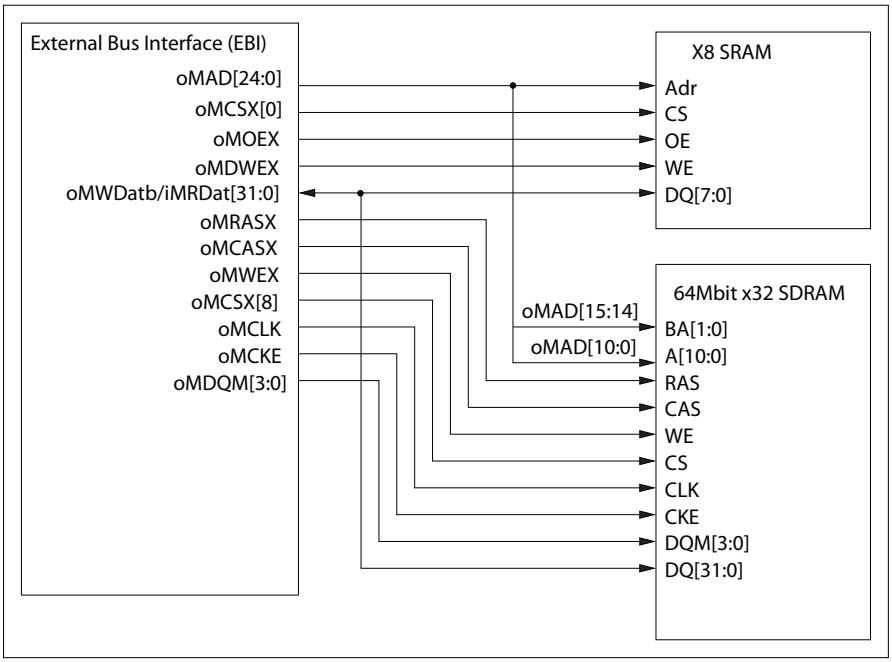
Note: Inputs/outputs in the above figures which end with an “X”, for example “oMOEX”, indicate active low inputs/outputs

Figure 42-24. Two 16-bit SRAM with External Bus Interface



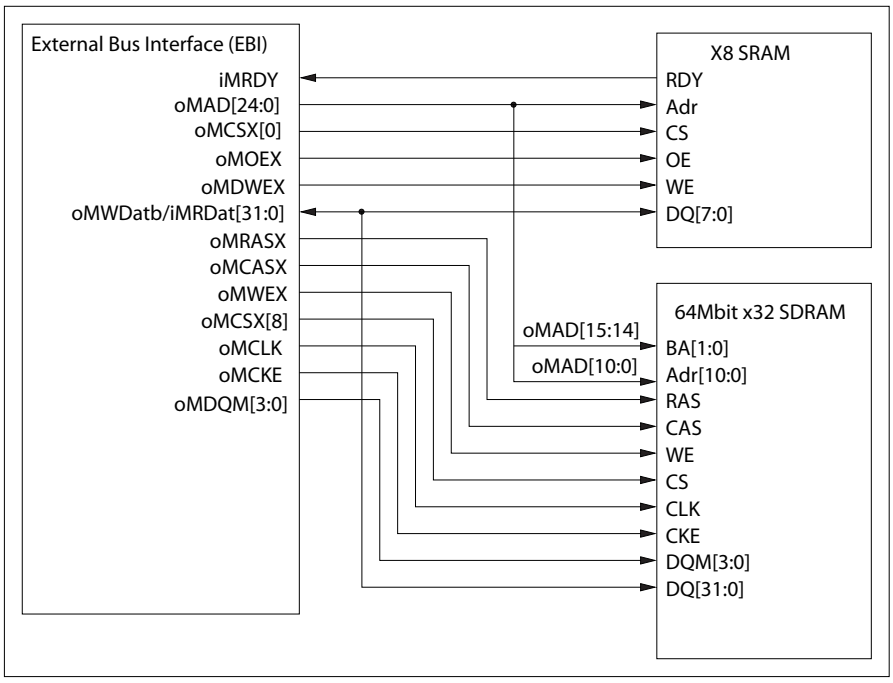
Note: Inputs/outputs in the above diagram which end with an “X”, for example “oMOEX”, indicate active low inputs/outputs

Figure 42-25. 8-bit SRAM and 32-bit SDRAM with External Bus Interface



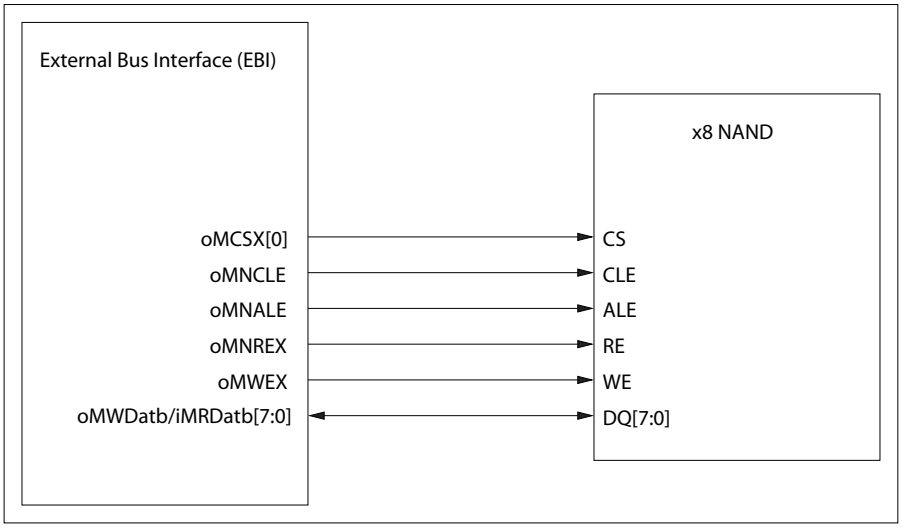
Note: Inputs/outputs in the above diagram which end with an “X”, for example “oMOEX”, indicate active low inputs/outputs

Figure 42-26. 32-bit SDRAM with low speed peripheral



Note: Inputs/outputs in the above diagram which end with an “X”, for example “oMOEX”, indicate active low inputs/outputs

Figure 42-27. 8-bit NAND with External Bus Interface IF



Note: Inputs/outputs in the above diagram which end with an “X”, for example “oMOEX”, indicate active low inputs/outputs

42.5 Access waveform examples for the External Bus Interface

This section gives examples showing access to various external memory.

Figure 42-28. Word read access to 8-bit width SRAM

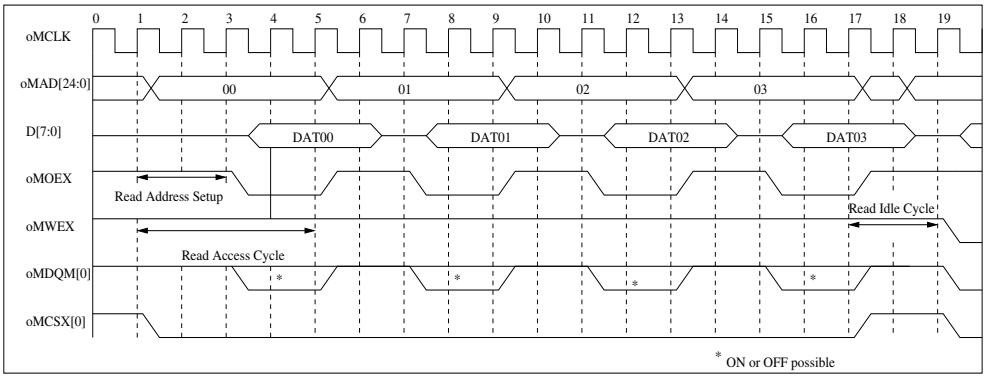


Figure 42-29. Word write/read continuous access to 16-bit width SRAM

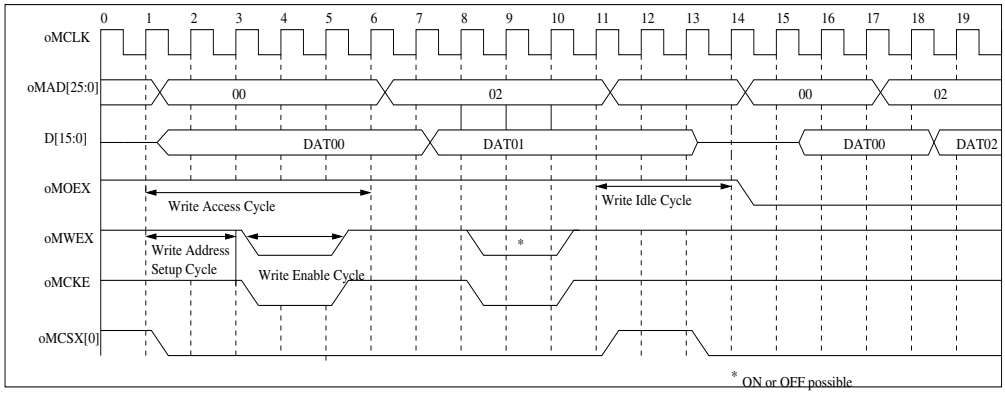


Figure 42-30. Read to low-speed device

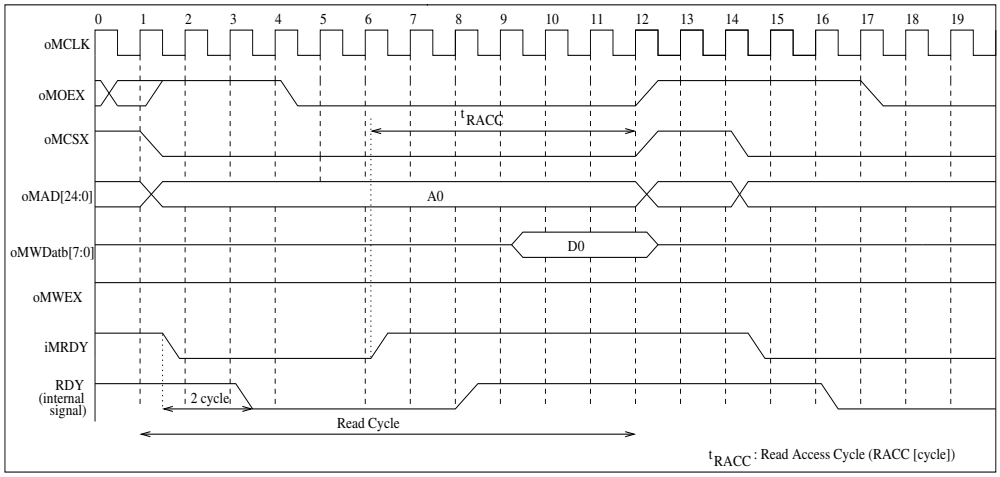


Figure 42-31. Write to low-speed device

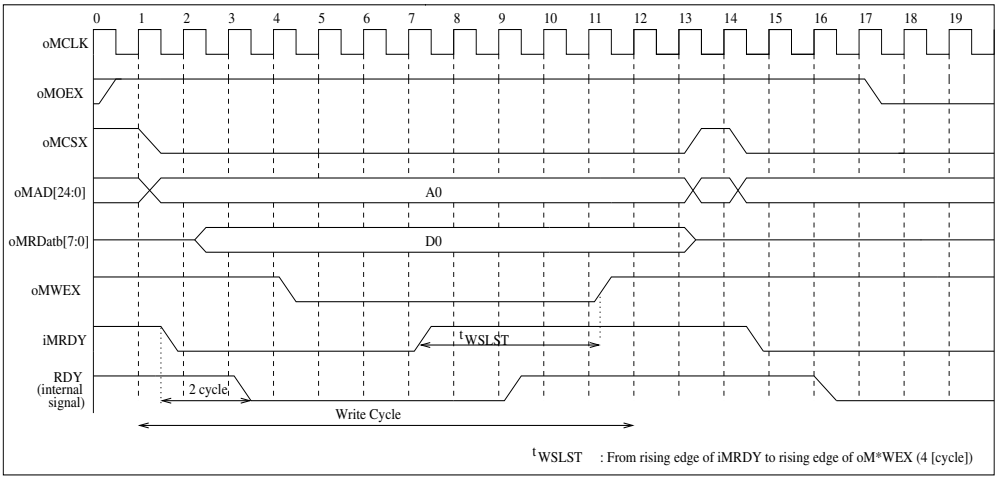


Figure 42-32. Read to low-speed device with split transfer on Memory Interface

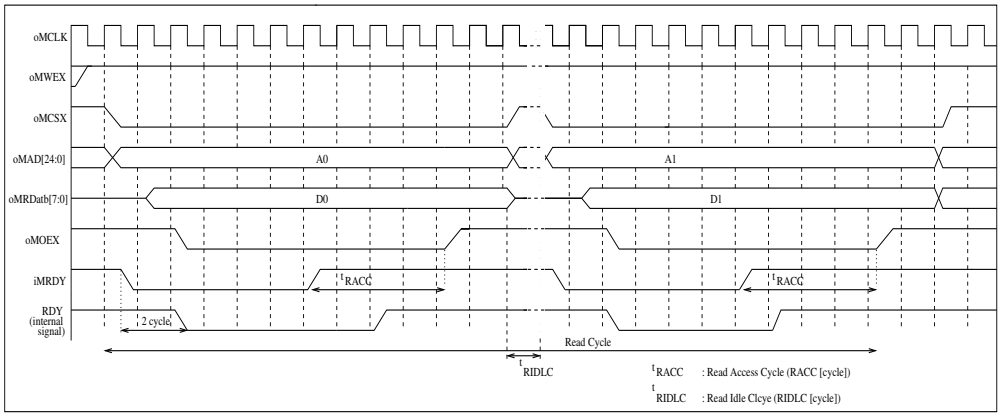


Figure 42-33. Write to low-speed device with split transfer on Memory Interface

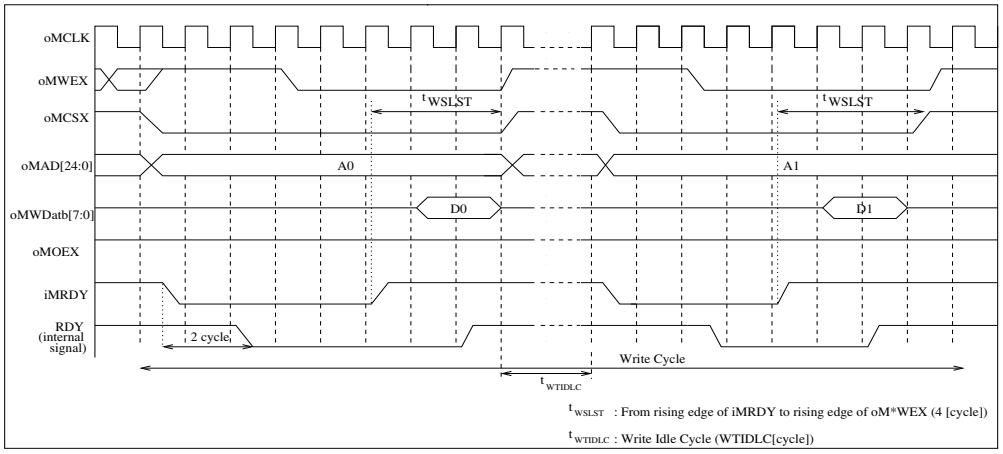


Figure 42-34. 8-bit NAND read or write command issue

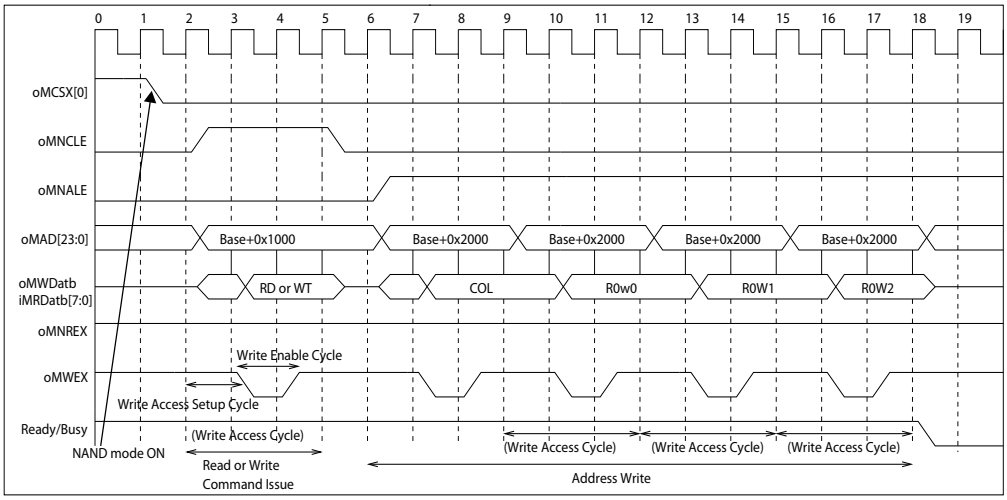


Figure 42-35. 8-bit NAND status read

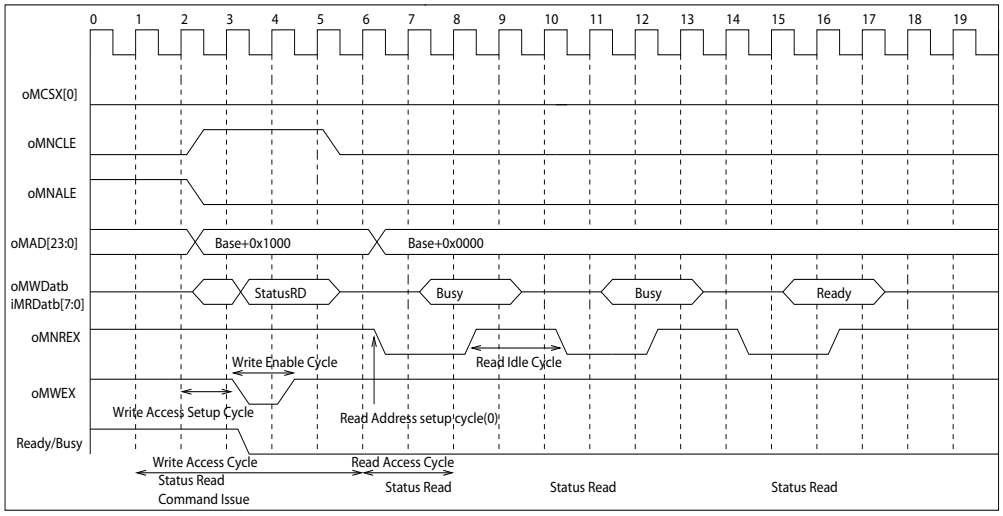


Figure 42-36. 8-bit NAND data write

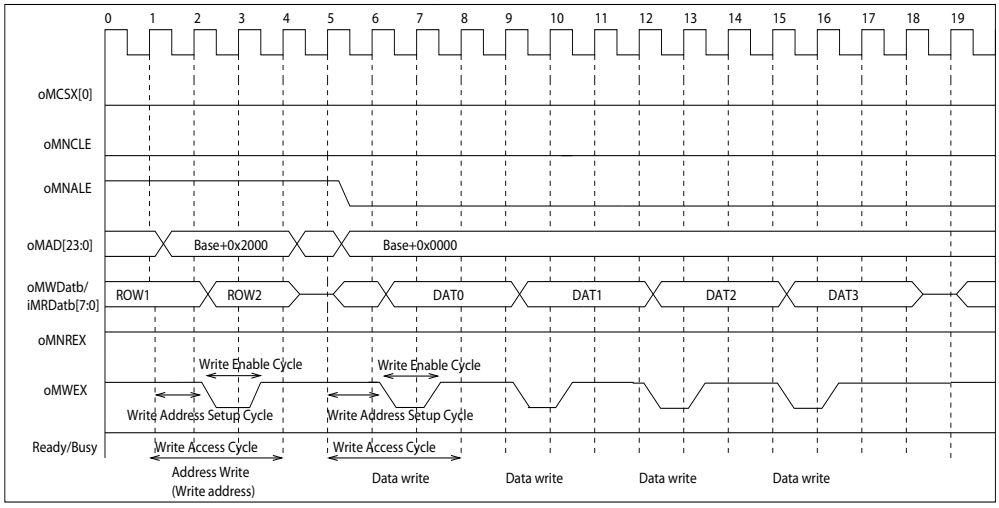


Figure 42-37. 16-bit NOR page read

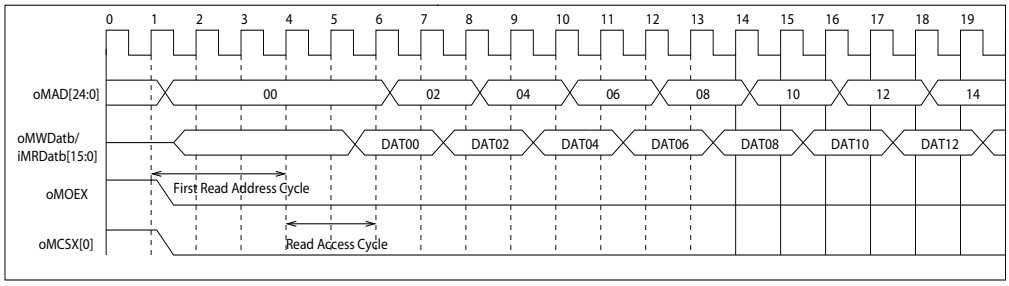


Figure 42-38. 32-bit SDRAM 32-byte word read access

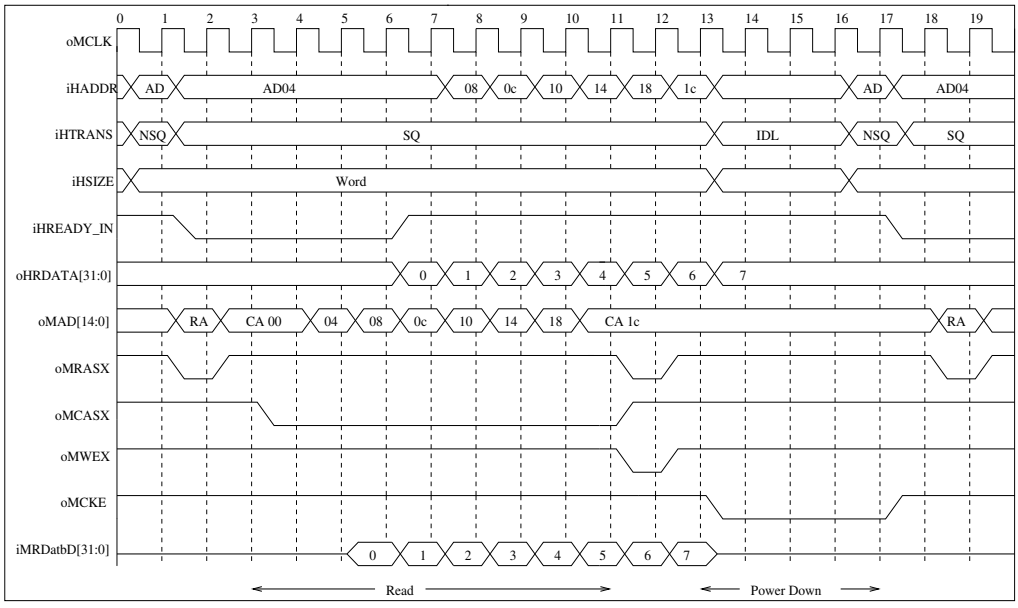
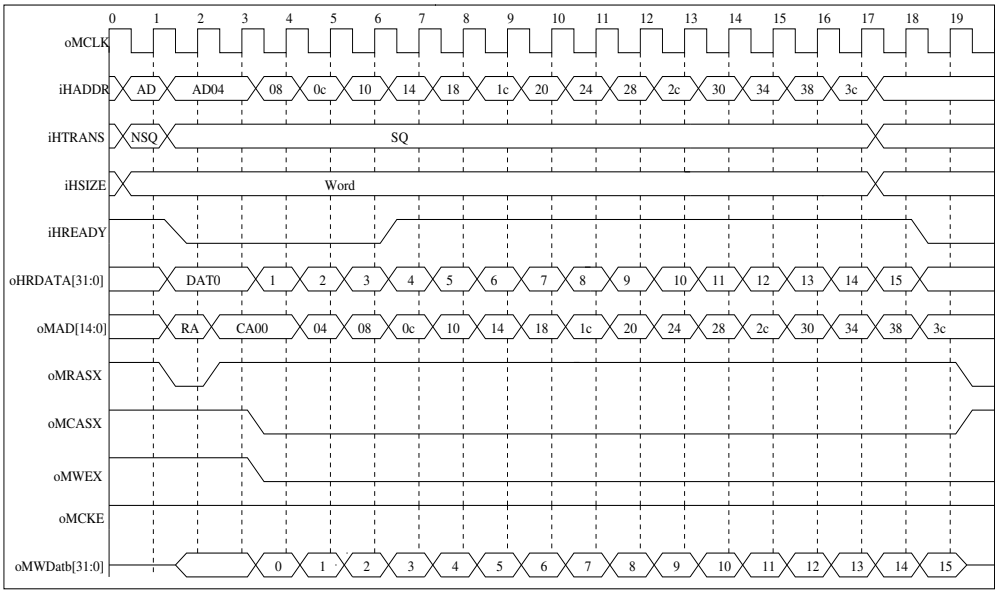


Figure 42-39. 32-bit SDRAM 64-byte word write access



Refer to the Device Datasheet for details on AC timings of EBI.

43. Secure Hardware Extension (SHE)

This chapter explains the function and operation of the SHE module.

43.1 Outline of the Secure Hardware Extension (SHE) module

This section describes the features and the block diagram of the Secure Hardware Extension (SHE) module.

Features of SHE

The Secure Hardware Extension (SHE) module is an on-chip peripheral of the microcontroller that provides cryptographic functions and secure key storage. This chapter describes the hardware part of the Cypress SHE implementation which, together with the software driver, fulfills the 'SHE Functional Specification v1.1, rev 439' [1] as well as Errata and amendments to this specification.

Features of the SHE module are:

- It implements all commands defined by the Functional Specification of SHE (7. Memory Protection Unit for AHB), such as
  - It provides AES-128 encryption and decryption operations [2] in Electronic Cipher Book (ECB) and Cipher-Block Chaining (CBC) modes [3]
  - It supports the generation of the cipher-based message authentication code (CMAC) [4]
  - it implements the Miyaguchi-Preneel compression function (see [5] Algorithm 9.43).

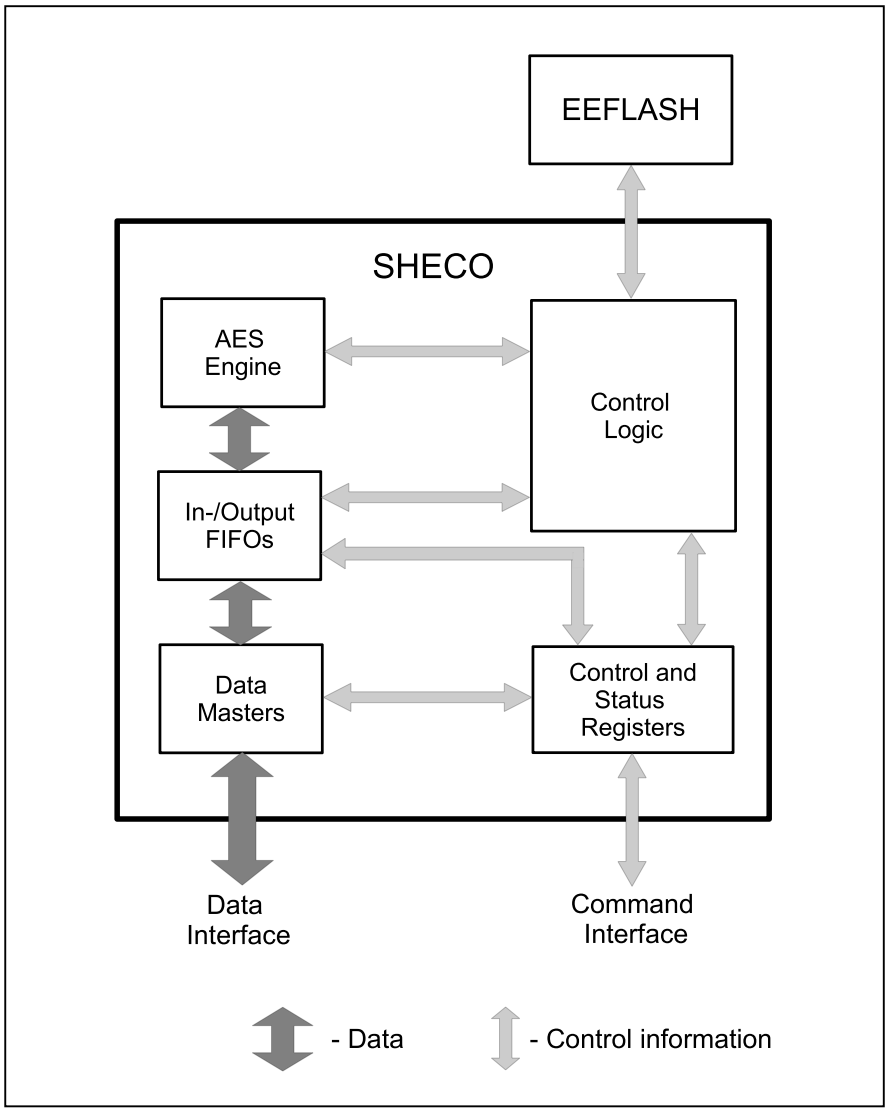
  Note: The Miyaguchi-Preneel compression function is not directly accessible by the user and will only be used internally if required during command processing
  - It provides a random number generation function
- SHE supports secure booting (See SHE Functional Specification, 10. EEPROM Emulation Flash) by ensuring:
  - Measurement before application start-up
  - Measurement during application start-up
  - Secure boot mode, start address and length of the bootloader are configurable by the user
- Secure key storage implemented in the EEFLASH (refer to 11. EEPROM Emulation Flash and SHE for more information). In this implementation:
  - Two small sectors are exclusively accessible from SHE and are not available for the user software (public side)
  - There is a round robin arbitration of accesses from SHE and the CPU
- It has a unique random and read-only SECRET_KEY inserted during production, which is only accessible inside the SHE module
- SHE has a unique read-only Identification item (UID), which is inserted during production
- It has a Command Interface for accessing configuration and status registers. This can also be used for data transfer to and from internal First-in/First-out (FIFOs)
- It has a data interface for direct access to microcontroller memory locations. This interface is:
  - Implemented as a master interface
  - Independent of read and write transfers
  - Configured by the software
- It implements FIFO-based data transfer for:
  - Command parameters

- Data to be processed. Only 128-bit data blocks can be processed by SHE, hence all padding has to be done by the application
- Results

- The internal SHE clock implements clock gating in order to reduce power consumption
- It features external debugger notification

Block diagram of the SHE module

Figure 43-1. Block diagram of the SHE module



43.2 Secure Hardware Extension registers

This section lists and describes all the SHE registers.

Registers of the SHE module

The following registers are available for the SHE module:

- Configuration registers for command interface
  - SHE Command Register (SHE_CMD)
  - SHE Command Cancel Register (SHE_CMDCANCEL)
  - SHE Clock Control Register (SHE_CLKCTRL)
- Status registers for command interface
  - SHE Status Register (SHE_STATUS)
  - SHE Error Code Register (SHE_ERC)
  - SHE Clock Status Register (SHE_CLKSTAT)
  - SHE Module ID Register (SHE_MID)
- Interrupt registers for command interface
  - Interrupt Request Register (SHE_IRQ)
  - Interrupt Request Enable Register (SHE_IRQEN)
  - Interrupt Request Clear Register (SHE_IRQCLR)
- Configuration registers for data interface
  - Input Channel Master Start Address Register (SHE_IMSTADDR)
  - Output Channel Master Start Address Register (SHE_OMSTADDR)
  - Input Channel Master Data Transfer Counter (SHE_IMSTCNT)
  - Output Channel Master Data Transfer Counter (SHE_OMSTCNT)
  - Input Channel Master Start Trigger (SHE_IMSTSTART)
  - Output Channel Master Start Trigger (SHE_OMSTSTART)
  - Input FIFO Configuration Register (SHE_IFIFOCFG)
  - Output FIFO Configuration Register (SHE_OFIFOCFG)
  - Input FIFO Compare Value Register (SHE_COMPARE0~1)
- Status registers for data interface
  - Compare Register Access Status Register (SHE_COMPACC)
  - Data Master Status Register (SHE_MSTSTATUS)
  - Input Channel Master Error Response Address Register (SHE_IMSTERRADDR)
  - Output Channel Master Error Response Address Register (SHE_OMSTERRADDR)
  - FIFO Status Register (SHE_FIFOSTATUS)
  - FIFO Load Register (SHE_FIFOLOAD)
  - Input FIFO Data Counter Register (SHE_DATACNT0~1)
- Data transfer registers
  - Input FIFO Write Data Register (SHE_IFIFOWRDATA0~31)
  - Output FIFO Read Data Register (SHE_OFIFORDDATA0~31)

Memory layout of SHE registers

Table 43-1. Memory layout of the SHE module registers

Offset	+3	+2	+1	+0
0x00000000	SHE_CMD 00000000 00000000 00000000 00000000			
0x00000004	SHE_CMDCANCEL 00000000 00000000 00000000 00000000			
0x00000008	SHE_CLKCTRL 00000000 00000000 00000000 00000000			
0x0000000C	SHE_STATUS 00000000 00000000 00000000 00000000			
0x00000010	SHE_ERC 00000000 00010000 00000000 00000000			
0x00000014	SHE_CLKSTAT 00000000 00000000 00000000 00000000			
0x00000018	SHE_MID 00000000 00000000 00000000 00000000			
0x0000001C	SHE_IRQ 00000000 00000000 00000000 00000000			
0x00000020	SHE_IRQEN 00000000 00000000 00000000 00000000			
0x00000024	SHE_IRQCLR 00000000 00000000 00000000 00000000			
0x00000028	SHE_IMSTADDR 00000000 00000000 00000000 00000000			
0x0000002C	SHE_OMSTADDR 00000000 00000000 00000000 00000000			
0x00000030	SHE_IMSTCNT 00000000 00000000 00000000 00000000			
0x00000034	SHE_OMSTCNT 00000000 00000000 00000000 00000000			
0x00000038	SHE_IMSTSTART 00000000 00000000 00000000 00000000			
0x0000003C	SHE_OMSTSTART 00000000 00000000 00000000 00000000			
0x00000040	SHE_IFIFOCFG 00000000 00000001 00000000 00000000			
0x00000044	SHE_OFIFOCFG 00000000 00000001 00000000 00000000			

Table 43-1. Memory layout of the SHE module registers

Offset	+3	+2	+1	+0
0x00000048	SHE_COMPARE0 11111111 11111111 11111111 11111111			
0x0000004C	SHE_COMPARE1 00000111 11111111 11111111 11111111			
0x00000050	SHE_COMPACC 00000000 00000000 00000000 00000000			
0x00000054	SHE_MSTSTATUS 00000000 00000001 00000000 00000001			
0x00000058	SHE_IMSTERRADDR 00000000 00000000 00000000 00000000			
0x0000005C	SHE_OMSTERRADDR 00000000 00000000 00000000 00000000			
0x00000060	SHE_FIFOSTATUS 00000000 00000000 00000000 00000000			
0x00000064	SHE_FIFOLOAD 00000000 00110000 00000000 00110000			
0x00000068	SHE_DATACNT0 00000000 00000000 00000000 00000000			
0x0000006C	SHE_DATACNT1 00000000 00000000 00000000 00000000			
0x00000070 - 0x000000FC	reserved 00000000 00000000 00000000 00000000			
0x00000100	SHE_IFIFOWRDATA0 00000000 00000000 00000000 00000000			
0x00000104	SHE_IFIFOWRDATA1 00000000 00000000 00000000 00000000			
0x00000108	SHE_IFIFOWRDATA2 00000000 00000000 00000000 00000000			
0x0000010C	SHE_IFIFOWRDATA3 00000000 00000000 00000000 00000000			
0x00000110	SHE_IFIFOWRDATA4 00000000 00000000 00000000 00000000			
0x00000114	SHE_IFIFOWRDATA5 00000000 00000000 00000000 00000000			
0x00000118	SHE_IFIFOWRDATA6 00000000 00000000 00000000 00000000			
0x0000011C	SHE_IFIFOWRDATA7 00000000 00000000 00000000 00000000			

Table 43-1. Memory layout of the SHE module registers

Offset	+3	+2	+1	+0
0x00000120	SHE_IFIFOWRDATA8 00000000 00000000 00000000 00000000			
0x00000124	SHE_IFIFOWRDATA9 00000000 00000000 00000000 00000000			
0x00000128	SHE_IFIFOWRDATA10 00000000 00000000 00000000 00000000			
0x0000012C	SHE_IFIFOWRDATA11 00000000 00000000 00000000 00000000			
0x00000130	SHE_IFIFOWRDATA12 00000000 00000000 00000000 00000000			
0x00000134	SHE_IFIFOWRDATA13 00000000 00000000 00000000 00000000			
0x00000138	SHE_IFIFOWRDATA14 00000000 00000000 00000000 00000000			
0x0000013C	SHE_IFIFOWRDATA15 00000000 00000000 00000000 00000000			
0x00000140	SHE_IFIFOWRDATA16 00000000 00000000 00000000 00000000			
0x00000144	SHE_IFIFOWRDATA17 00000000 00000000 00000000 00000000			
0x00000148	SHE_IFIFOWRDATA18 00000000 00000000 00000000 00000000			
0x0000014C	SHE_IFIFOWRDATA19 00000000 00000000 00000000 00000000			
0x00000150	SHE_IFIFOWRDATA20 00000000 00000000 00000000 00000000			
0x00000154	SHE_IFIFOWRDATA21 00000000 00000000 00000000 00000000			
0x00000158	SHE_IFIFOWRDATA22 00000000 00000000 00000000 00000000			
0x0000015C	SHE_IFIFOWRDATA23 00000000 00000000 00000000 00000000			
0x00000160	SHE_IFIFOWRDATA24 00000000 00000000 00000000 00000000			
0x00000164	SHE_IFIFOWRDATA25 00000000 00000000 00000000 00000000			
0x00000168	SHE_IFIFOWRDATA26 00000000 00000000 00000000 00000000			

Table 43-1. Memory layout of the SHE module registers

Offset	+3	+2	+1	+0
0x0000016C	SHE_IFIFOWRDATA27 00000000 00000000 00000000 00000000			
0x00000170	SHE_IFIFOWRDATA28 00000000 00000000 00000000 00000000			
0x00000174	SHE_IFIFOWRDATA29 00000000 00000000 00000000 00000000			
0x00000178	SHE_IFIFOWRDATA30 00000000 00000000 00000000 00000000			
0x0000017C	SHE_IFIFOWRDATA31 00000000 00000000 00000000 00000000			
0x00000180	SHE_OFIFORDDATA0 00000000 00000000 00000000 00000000			
0x00000184	SHE_OFIFORDDATA1 00000000 00000000 00000000 00000000			
0x00000188	SHE_OFIFORDDATA2 00000000 00000000 00000000 00000000			
0x0000018C	SHE_OFIFORDDATA3 00000000 00000000 00000000 00000000			
0x00000190	SHE_OFIFORDDATA4 00000000 00000000 00000000 00000000			
0x00000194	SHE_OFIFORDDATA5 00000000 00000000 00000000 00000000			
0x00000198	SHE_OFIFORDDATA6 00000000 00000000 00000000 00000000			
0x0000019C	SHE_OFIFORDDATA7 00000000 00000000 00000000 00000000			
0x000001A0	SHE_OFIFORDDATA8 00000000 00000000 00000000 00000000			
0x000001A4	SHE_OFIFORDDATA9 00000000 00000000 00000000 00000000			
0x000001A8	SHE_OFIFORDDATA10 00000000 00000000 00000000 00000000			
0x000001AC	SHE_OFIFORDDATA11 00000000 00000000 00000000 00000000			
0x000001B0	SHE_OFIFORDDATA12 00000000 00000000 00000000 00000000			
0x000001B4	SHE_OFIFORDDATA13 00000000 00000000 00000000 00000000			

Table 43-1. Memory layout of the SHE module registers

Offset	+3	+2	+1	+0
0x000001B8	SHE_OFIFORDDATA14 00000000 00000000 00000000 00000000			
0x000001BC	SHE_OFIFORDDATA15 00000000 00000000 00000000 00000000			
0x000001C0	SHE_OFIFORDDATA16 00000000 00000000 00000000 00000000			
0x000001C4	SHE_OFIFORDDATA17 00000000 00000000 00000000 00000000			
0x000001C8	SHE_OFIFORDDATA18 00000000 00000000 00000000 00000000			
0x000001CC	SHE_OFIFORDDATA19 00000000 00000000 00000000 00000000			
0x000001D0	SHE_OFIFORDDATA20 00000000 00000000 00000000 00000000			
0x000001D4	SHE_OFIFORDDATA21 00000000 00000000 00000000 00000000			
0x000001D8	SHE_OFIFORDDATA22 00000000 00000000 00000000 00000000			
0x000001DC	SHE_OFIFORDDATA23 00000000 00000000 00000000 00000000			
0x000001E0	SHE_OFIFORDDATA24 00000000 00000000 00000000 00000000			
0x000001E4	SHE_OFIFORDDATA25 00000000 00000000 00000000 00000000			
0x000001E8	SHE_OFIFORDDATA26 00000000 00000000 00000000 00000000			
0x000001EC	SHE_OFIFORDDATA27 00000000 00000000 00000000 00000000			
0x000001F0	SHE_OFIFORDDATA28 00000000 00000000 00000000 00000000			
0x000001F4	SHE_OFIFORDDATA29 00000000 00000000 00000000 00000000			
0x000001F8	SHE_OFIFORDDATA30 00000000 00000000 00000000 00000000			
0x000001FC	SHE_OFIFORDDATA31 00000000 00000000 00000000 00000000			

43.2.1 Configuration registers for the Command Interface

43.2.1.1 SHE Command Register (SHE_CMD)

The SHE command register is used to initiate the execution of sequential commands, i.e. all commands defined in 7. Memory Protection Unit for AHB of the SHE Functional Specification except for CMD_CANCEL and CMD_GET_STATUS.

Writing valid command opcode to this register starts command execution. Writing invalid command opcode generates an ERC_GENERAL_ERROR error code in the error register bit field SHE_ERC:CMDMAIN.

The busy flag (SHE_STATUS:BUSY) is set and the done flag (SHE_STATUS:DONE) is cleared by the hardware with a write access to the SHE_CMD register. Moreover, the error code register SHE_ERC:CMDMAIN is set to ERC_INVALID to indicate that error code is not yet available.

A new sequential command can only be started if the previous one has finished. Hence the register is locked for write accesses while command execution is ongoing, i.e. SHE_STATUS:BUSY is '1'. Any write access will trigger a bus error response. Moreover, the command register SHE_CMD is also locked during the execution of the command CMD_CANCEL, i.e. while SHE_STATUS:BUSY is high. Refer to 43.2.2.1 SHE Status Register (SHE_STATUS) for more information about using this register.

SHE Command Register (SHE_CMD)

Figure 43-2. SHE Command Register (SHE_CMD)

SHE_CMD																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CMD[7]	CMD[6]	CMD[5]	CMD[4]	CMD[3]	CMD[2]	CMD[1]	CMD[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-2. SHE Command Register (SHE_CMD) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-

Table 43-2. SHE Command Register (SHE_CMD) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	CMD	Command Opcode register Listed below are possible opcode values with the related commands. Other values are reserved and will generate an ERC_GENERAL_ERROR error code in the SHE_ERC:CMDMAIN error register. 0x01: CMD_ENCRYPT_ECB 0x02: CMD_ENCRYPT_CBC 0x03: CMD_DECRYPT_ECB 0x04: CMD_DECRYPT_CBC 0x05: CMD_GENERATE_MAC 0x06: CMD_VERIFY_MAC 0x07: CMD_LOAD_KEY 0x08: CMD_LOAD_PLAIN_KEY 0x09: CMD_EXPORT_RAM_KEY 0x0A: CMD_INIT_RNG 0x0B: CMD_EXTEND_SEED 0x0C: CMD_RND 0x0D: CMD_SECURE_BOOT 0x0E: CMD_BOOT_FAILURE 0x0F: CMD_BOOT_OK 0x11: CMD_GET_ID 0x13: CMD_DEBUG **Notes:** 1. The concurrent command CMD_CANCEL is started using its own register (SHE_CMDCANCEL). 2. The concurrent command CMD_GET_STATUS simply reads the SHE status register (SHE_STATUS).

43.2.1.2 SHE Command Cancel Register (SHE_CMDCANCEL)

This is a dedicated register that initiates the CMD_CANCEL command concurrently with another ongoing command.

The busy flag (SHE_STATUS:BUSY) is set and done flag (SHE_STATUS:DONE) is cleared by the hardware upon write access to the SHE_CMDCANCEL register. Moreover, the error code for the CMD_CANCEL command (SHE_ERC:CANCEL-MAIN) is set to ERC_INVALID. Note: The SHE command register (SHE_CMD) is locked while the CMD_CANCEL command is being processed.

SHE Command Cancel Register (SHE_CMDCANCEL)

Figure 43-3. SHE Command Cancel Register (SHE_CMDCANCEL)

SHE_CMDCANCEL																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CANCELREQ
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-3. SHE Command Cancel Register (SHE_CMDCANCEL) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	CANCELREQ	Cancel Request bit Any ongoing command execution inside SHE can be canceled by writing this bit to '1'. It remains '1' while the cancel request is being processed and will be cleared by the SHE control logic afterwards.

43.2.1.3 SHE Clock Control Register (SHE_CLKCTRL)

This register is used for enabling/disabling the clock. The software can disable the internal clock of the SHE module in order to reduce power consumption.

Note: Only the SHE_CLKCTRL and SHE_CLKSTAT registers are accessible if the internal SHE clock is disabled. Accessing other SHE registers will generate a bus error response. Ref er to Clock gating for SHE for more details.

SHE Clock Control Register (SHE_CLKCTRL)

Figure 43-4. SHE Clock Control Register (SHE_CLKCTRL)

SHE_CLKCTRL																														
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	DISREQ	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	ENREQ
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp1	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-4. SHE Clock Control Register (SHE_CLKCTRL) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	DISREQ	Clock Disable Request bit Writing this bit to '1' triggers a clock disable request to the SHE control logic. DISREQ is automatically cleared when the SHE clock is disabled (i.e. SHE_CLKSTAT:CLKOFF = '1'). The internal SHE clock can only be disabled if SHE is in idle mode, i.e. if the clock disable request is written during ongoing command execution, then the clock won't be disabled until SHE becomes idle. This bit can only be written to '1'. Writing '0' is ignored. Writing '1' while the SHE clock is disabled is ignored and the DISREQ bit remains '0'.
[15:8]	read0	-
[7:1]	read0	-

Table 43-4. SHE Clock Control Register (SHE_CLKCTRL) bits

Bit Position	Bit Field Name	Bit Description
[0]	ENREQ	Clock Enable Request bit Writing this bit to '1' triggers a clock enable request to the SHE control logic. ENREQ is automatically cleared when the SHE clock is enabled (i.e. SHE_CLKSTAT:CLKOFF = '0'). This bit can only be written to '1'. Writing '0' is ignored. Writing '1' while the SHE clock is enabled is ignored and the ENREQ bit remains '0'.

43.2.2 Status registers for the command interface

43.2.2.1 SHE Status Register (SHE_STATUS)

A status register provides the state of SHE. It contains eight status bits, which are defined in the SHE Functional Specification (6. Memory Protection Unit for AXI) along with additional status bits required for more detailed information.

SHE Status Register (SHE_STATUS)

Figure 43-5. SHE Status Register (SHE_STATUS)

SHE_STATUS																															
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**
read0	read0	read0	read0	read0	read0	read0	FATALERR	read0	read0	read0	read0	FLASHDED	FLASHSEC	RAMDED	RAMSEC	read0	read0	read0	read0	read0	read0	INITDONE	DONE	INTDEBUGGER	EXTDEBUGGER	RNDINIT	BOOTOK	BOOTFINISHED	BOOTINIT	SECUREBOOT	BUSY
R0	R0	R0	R0	R0	R0	R0	Rp	R0	R0	R0	R0	Rp	Rp	Rp	Rp	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-5. SHE Status Register (SHE_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[31:28]	read0	-
[27:25]	read0	-
[24]	FATALERR	Fatal Error flag This flag is set in the event of an unrecoverable error inside the SHE module. It is cleared by a reset only. If an unrecoverable error is detected, SHE immediately stops its current operation, moves to a safe state and becomes unresponsive. A reset is required before SHE can be used again. **Note:** The update of the error code register SHE_ERC may take up to 100 clock cycles after the rise of the FATALERR flag, depending on the severity of the error.
[23:20]	read0	-
[19]	FLASHDED	Flash Double Bit ECC Error flag This flag indicates that a double bit Error Correction Code (ECC) error occurred in SHE's flash memory (dedicated part of EEFLASH). The flag is set on the first occurrence of the error and can only be cleared by a reset.

Table 43-5. SHE Status Register (SHE_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[18]	FLASHSEC	Flash Single Bit ECC Warning flag This flag indicates that a single bit ECC error occurred in SHE's flash memory (i.e. a dedicated part of EEFLASH) and has been corrected. The flag is set on the first occurrence of the error and can only be cleared by a reset.
[17]	RAMDED	RAM Double Bit ECC Error flag This flag indicates that a double bit ECC error occurred in SHE's internal RAM. The flag is set on the first occurrence of the error and can only be cleared by a reset.
[16]	RAMSEC	RAM Single Bit ECC Warning flag This flag indicates that a single bit ECC error in SHE's internal RAM occurred and has been corrected. The flag is set on the first occurrence of the error and can only be cleared by a reset.
[15:10]	read0	-
[9]	INITDONE	SHE Initialization status flag This flag indicates that the initialization phase of SHE is finished.
[8]	DONE	Done status flag This flag indicates that command processing inside SHE is finished and an appropriate error code is present in the error code register SHE_ERC. It is cleared immediately upon a 32-bit read access to SHE_ERC.
[7]	INTDEBUGGER	Internal Debugger status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set if the internal debugging mechanisms of SHE are activated. It is cleared by the reset only.
[6]	EXTDEBUGGER	External Debugger status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set if an external debugger is connected to the MCU, i.e. it reflects the input to activate the debugger. It is cleared by the reset only. **Note:** External debugger detection is active even if the SHE clock is switched off. More precisely, if an external debugger is attached while SHE is in power saving mode, then this will be detected and the EXTDEBUGGER bit set after the SHE clock has been enabled.

Table 43-5. SHE Status Register (SHE_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[5]	RNDINIT	Random Seed Initialization status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set if the random number generator has been initialized. It is cleared after successful authentication during the CMD_DEBUG command.
[4]	BOOTOK	Boot OK status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set if secure booting (CMD_SECURE_BOOT command) has succeeded. If the CMD_BOOT_FAILURE command is called, then the bit is erased. The Boot OK status flag is also cleared if some parts of the MCU enter power saving mode (refer to the 43.3.1.1 Power saving functionality for more information).
[3]	BOOTFINISHED	Boot Finished status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set when secure booting has been completed by calling either the CMD_BOOT_FAILURE or CMD_BOOT_OK commands (in the SHE_CMD register), or if the CMD_SECURE_BOOT command failed while verifying BOOT_MAC.
[2]	BOOTINIT	Boot Initialization status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set if secure booting has been personalised during the boot sequence.
[1]	SECUREBOOT	Secure Boot Activated status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set if secure booting is activated.

Table 43-5. SHE Status Register (SHE_STATUS) bits

Bit Position	Bit Field Name	Bit Description
[0]	BUSY	Busy status flag Defined in the SHE Functional Specification (6. Memory Protection Unit for AXI). See the 'References' section of this chapter. This bit is set whenever SHE is processing a command. The BUSY status flag is set immediately upon the start of a new command, i.e. write access to the command register (SHE_CMD) while BUSY is '0' or write access to the command cancel register SHE_CMDCANCEL. The BUSY status flag is cleared immediately upon a 32-bit read access to the error code register (SHE_ERC) while the SHE_STATUS:DONE flag is '1'.

43.2.2.2 SHE Error Code Register (SHE_ERC)

The SHE module outputs error code after the execution of a command in the SHE_ERC register. This register is divided into two parts: error codes for regular/sequential commands (all SHE commands except for CMD_CANCEL and CMD_GET_STATUS) and error codes for CMD_CANCEL.

The CMD_CANCEL command requires a dedicated error code register because it can be executed concurrently with other commands and also returns an error code. The currently executed command will return ERC_CANCELED to inform the software about its cancellation.

SHE Error Code Register (SHE_ERC)

Figure 43-6. SHE Error Code Register (SHE_ERC)

SHE_ERC																															
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**
CANCELEXTD[7]	CANCELEXTD[6]	CANCELEXTD[5]	CANCELEXTD[4]	CANCELEXTD[3]	CANCELEXTD[2]	CANCELEXTD[1]	CANCELEXTD[0]	CANCELMAIN[7]	CANCELMAIN[6]	CANCELMAIN[5]	CANCELMAIN[4]	CANCELMAIN[3]	CANCELMAIN[2]	CANCELMAIN[1]	CANCELMAIN[0]	CMDEXTD[7]	CMDEXTD[6]	CMDEXTD[5]	CMDEXTD[4]	CMDEXTD[3]	CMDEXTD[2]	CMDEXTD[1]	CMDEXTD[0]	CMDMAIN[7]	CMDMAIN[6]	CMDMAIN[5]	CMDMAIN[4]	CMDMAIN[3]	CMDMAIN[2]	CMDMAIN[1]	CMDMAIN[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-6. SHE Error Code Register (SHE_ERC) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	CANCELEXTD	Extended Command CMD_CANCEL Execution Error Code These bits contain the extended error code to the error code stored in the eight CANCELMAIN bits and is only valid if the value of the CANCELMAIN bits is equal to ERC_GENERAL_ERROR. It can be used to retrieve additional information about errors that have occured during the processing of CMD_CANCEL. 0x00: ERC_INVALID CMD_CANCEL processing ongoing; error code not yet available

Table 43-6. SHE Error Code Register (SHE_ERC) bits

Bit Position	Bit Field Name	Bit Description
[23:16]	CANCELMAIN	Command CMD_CANCEL Execution Error Code These bits contain the dedicated error code for the CMD_CANCEL command. The default value is ERC_CANCEL_IDLE and is written after a reset and after the start of a new command (i.e. a write access to the SHE_CMD register while SHE is idle). The value changes to ERC_INVALID immediately after a cancel request (i.e. a write access to SHE_CMD-CANCEL) and will be updated with the correct error code when CMD_CANCEL has finished processing. 0x00: ERC_INVALID CMD_CANCEL processing ongoing; error code not yet available 0x01: ERC_NO_ERROR CMD_CANCEL processing has completed without error 0x0D: ERC_GENERAL_ERROR A general error occurred during CMD_CANCEL processing 0x10: ERC_CANCEL_IDLE (Default value) No CMD_CANCEL processing is ongoing
[15:8]	CMDEXTD	Extended SHE Command Execution Error Code These bits contain the extended error code to the error code already stored in the eight CMDMAIN bits. The extended error code is only valid if the value of CMDMAIN is not equal to ERC_INVALID or ERC_NO_ERROR. It can be used to retrieve additional information about errors that have occurred. 0x00: ERC_INVALID The command is ongoing and the error code is not available, or the command has successfully finished (depending on value of CMDMAIN) 0x80: ERC_EXCESS_INPUT_DATA The input FIFO was not empty at the moment when the compare match event occurred

Table 43-6. SHE Error Code Register (SHE_ERC) bits

Bit Position	Bit Field Name	Bit Description
[7:0]	CMDMAIN	SHE Command Execution Error Code These bits contains the error codes which can be returned for the execution of all SHE commands with the exception of CMD_CANCEL and CMD_GET_STATUS. Two error code values are added and will be handled by the SHE software driver. Other values are reserved. The ERC_BUSY error code (not listed below) must be generated by the software driver if it receives an error response while writing to the locked command register (SHE_CMD). Upon the start of a new command, the error code is reset to ERC_INVALID until after the command has executed, when it is then changed to an appropriate value, e.g. it will be set to ERC_CANCELED if CMD_CANCEL was issued during command execution. Possible error code values, defined in the SHE Functional Specification (8. Programmable CRC) are summarized below: 0x00: ERC_INVALID The current command is still ongoing and error code is not available (Default value) 0x01: ERC_NO_ERROR No error has occurred and the command will be executed 0x02: ERC_SEQUENCE_ERROR Sequence of commands or subcommands is out of order 0x03: ERC_KEY_NOT_AVAILABLE A key is locked due to either a failed boot measurement or an active debugger 0x04: ERC_KEY_INVALID A function is called to perform an operation with a key that is not permitted for the given operation 0x05: ERC_KEY_EMPTY The application attempts to use a key not yet initialized 0x06: ERC_NO_SECURE_BOOT Returned by the CMD_SECURE_BOOT command when the conditions for a secure boot process have not been met and the process has to be canceled. It is also returned if a function changing the boot status is called without secure booting or after the secure boot process has finished 0x07: ERC_KEY_WRITE_PROTECTED A key update is attempted on a memory slot that has been write protected or when an attempt to active the debugger is started when a key is write-protected 0x08: ERC_KEY_UPDATE_ERROR A key update did not succeed due to errors in verifying the messages 0x09: ERC_RNG_SEED Returned by the CMD_RND and CMD_DEBUG commands if the seed has not been initialized 0x0A: ERC_NO_DEBUGGING Internal debugging is not possible because authentication with the challenge response protocol has not succeeded 0x0C: ERC_MEMORY_FAILURE See 8. Programmable CRC in the SHE Functional Specification 0x0D: ERC_GENERAL_ERROR See 8. Programmable CRC in the SHE Functional Specification 0x0E: ERC_CANCELED Command execution has been canceled

43.2.2.3 SHE Clock Status Register (SHE_CLKSTAT)

This register provides information about whether the SHE module clock is enabled or disabled in order to reduce power consumption.

SHE Clock Status Register (SHE_CLKSTAT)

Figure 43-7. SHE Clock Status Register (SHE_CLKSTAT)

SHE_CLKSTAT																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CLKOFF
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-7. SHE Clock Status Register (SHE_CLKSTAT) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	CLKOFF	Clock Disabled flag This bit is set to '1' if the SHE clock is disabled. Otherwise it is cleared

43.2.2.4 SHE Module ID Register (SHE_MID)

This is a read-only register with a unique module identification number (MID) which identifies the SHE module version used in the MCU.

SHE Module ID Register (SHE_MID)

Figure 43-8. SHE Module ID Register (SHE_MID)

SHE_MID																															
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**
MID[31]	MID[30]	MID[29]	MID[28]	MID[27]	MID[26]	MID[25]	MID[24]	MID[23]	MID[22]	MID[21]	MID[20]	MID[19]	MID[18]	MID[17]	MID[16]	MID[15]	MID[14]	MID[13]	MID[12]	MID[11]	MID[10]	MID[9]	MID[8]	MID[7]	MID[6]	MID[5]	MID[4]	MID[3]	MID[2]	MID[1]	MID[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-8. SHE Module ID Register (SHE_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID register This register identifies the particular version of the hardware used in the device. **Note:** Refer to the device specific datasheet for more details about the module identification number.

43.2.3 Interrupt registers for command interface

43.2.3.1 Interrupt Request Register (SHE_IRQ)

This register contains active interrupt requests, each of which is associated with a specific event. The lower half-word represents regular interrupts, i.e. events which are expected during command execution. All these interrupts are combined and propagated to the interrupt controller on a single shared interrupt line.

The upper half-word represents error interrupts. These interrupts are also combined, resulting in a shared error interrupt line. All interrupts can be enabled and cleared independently of each other using the SHE_IRQEN and SHE_IRQCLR registers respectively. Refer to Interrupt request generation for more information on interrupt generation/processing.

Interrupt Request Register (SHE_IRQ)

Figure 43-9. Interrupt Request Register (SHE_IRQ)

SHE_IRQ																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	FATALERR	IFIFOLOCKERR	OMSTIFSELERR	IMSTIFSELERR	OMSTERR	IMSTERR	OFIFORDERR	IFIFOWRERR	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	OFIFORDTH	IFIFOWRTH	OMSTIDLE	IMSTIDLE	DONE	COMPAREMATCH
R0	R0	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-9. Interrupt Request Register (SHE_IRQ) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	FATALERR	Fatal Error Interrupt flag This bit is set in the event of an unrecoverable error inside the SHE module (on the rising edge of SHE_STATUS:FATALERR). The Fatal Error Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:FATALERR bit).
[22]	IFIFOLOCKERR	Write To Locked Input FIFO Interrupt flag This bit is set if a write access to the locked input FIFO is performed either by the input channel master or through the command interface. The Write To Locked Input FIFO Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:IFIFOLOCKERR bit).

Table 43-9. Interrupt Request Register (SHE_IRQ) bits

Bit Position	Bit Field Name	Bit Description
[21]	OMSTIFSELERR	Output Data Channel Interface Selection Error Interrupt flag This bit is set if a data transfer start for the output data channel master is requested (i.e. write access to the SHE_OMSTSTART register while in idle state) and the output FIFO interface selection is configured to 'Command Interface'. The Output Data Channel Interface Selection Error Interrupt flag can only be cleared by the software (refer to the SHE_IRQCLR:OMSTIFSELERR bit).
[20]	IMSTIFSELERR	Input Data Channel Interface Selection Error Interrupt flag This bit is set if a data transfer start for the input data channel master is requested (i.e. write access to the SHE_IMSTSTART register while in idle state) and the input FIFO interface selection is configured to 'Command Interface'. The Input Data Channel Interface Selection Error Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:IMSTIFSELERR bit).
[19]	OMSTERR	Output Channel Master Error Interrupt flag This bit is set if the output channel master receives an error response on the AXI bus (on the rising edge of SHE_MSTSTATUS:OMSTERR). The Ouput Channel Master Error Interrupt flag can only be cleared by the software (refer to the SHE_IRQCLR:OMSTERR bit).
[18]	IMSTERR	Input Channel Master Error Interrupt flag This bit is set if the input channel master receives an error response on the AXI bus (on the rising edge of SHE_MSTSTATUS:IMSTERR). The Input Channel Master Error Interrupt flag can only be cleared by the software (refer SHE_IRQCLR:IMSTERR bit).
[17]	OFIFORDERR	Read From Empty Output FIFO Error Interrupt flag This bit is set if a read access to the empty output First-in First-out (FIFO) is performed over the Command Interface. The Read From Empty Output FIFO Error Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:OFIFORDERR bit).
[16]	IFIFOWRERR	Write To Full Input FIFO Error Interrupt flag This bit is set if a write access to the full input FIFO is performed over the Command Interface. The Write To Full Input FIFO Error Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:IFIFOWRERR bit).
[15:8]	read0	-

Table 43-9. Interrupt Request Register (SHE_IRQ) bits

Bit Position	Bit Field Name	Bit Description
[7:6]	read0	-
[5]	OFIFORDTH	Output FIFO Above Threshold Interrupt flag This bit is set if the amount of data words stored in the output FIFO becomes greater than the configured read threshold value (on the rising edge of SHE_FIFOSTATUS:OFIFORDTH). The Output FIFO Above Threshold Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:OFIFORDTH bit).
[4]	IFIFOWRTH	Input FIFO Below Threshold Interrupt flag This bit is set if the amount of data words stored in the input FIFO is less than the configured write threshold value (on the rising edge of SHE_FIFOSTATUS:IFIFOWRTH). The Input FIFO Below Threshold Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:IFIFOWRTH bit).
[3]	OMSTIDLE	Output Channel Master Idle Interrupt flag This bit is set if the output channel master moves to the idle state (on the rising edge of SHE_MSTSTATUS:OMSTIDLE). The Output Channel Master Idle Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:OMSTIDLE bit).
[2]	IMSTIDLE	Input Channel Master Idle Interrupt flag This bit is set if the input channel master moves to the idle state (on the rising edge of SHE_MSTSTATUS:IMSTIDLE). The Input Channel Master Idle Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:IMSTIDLE bit).
[1]	DONE	Command Execution Done Interrupt flag This bit is set if SHE internal processing has finished (on the rising edge of SHE_STATUS:DONE). The Command Execution Done Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:DONE bit).
[0]	COMPAREMATCH	Compare Match Interrupt flag This bit is set if the required amount of data has been transferred through the input FIFO (on the rising edge of SHE_FIFOSTATUS:COMPAREMATCH). The Compare Match Interrupt flag can only be cleared by software (refer to the SHE_IRQCLR:COMPAREMATCH bit).

43.2.3.2 Interrupt Request Enable Register (SHE_IRQEN)

This register enables the signalling of interrupts stored in SHE_IRQ to the host CPU.

Refer to Interrupt request generation for more information on interrupt generation/processing.

Interrupt Request Enable Register (SHE_IRQEN)

Figure 43-10. Interrupt Request Enable Register (SHE_IRQEN)

SHE_IRQEN																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	FATALERR	IFIFOLOCKERR	OMSTIFSELERR	IMSTIFSELERR	OMSTERR	IMSTERR	OFIFORDERR	IFIFOWRERR	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	OFIFORDTH	IFIFOWRTH	OMSTIDLE	IMSTIDLE	DONE	COMPAREMATCH
R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-10. Interrupt Request Enable Register (SHE_IRQEN) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	FATALERR	Fatal Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:FATALERR will be propagated to the interrupt controller.
[22]	IFIFOLOCKERR	Write To Locked Input FIFO Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:IFIFOLOCK-ERR will be propagated to the interrupt controller.
[21]	OMSTIFSELERR	Output Data Channel Interface Selection Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:OMSTIFSEL-ERR will be propagated to the interrupt controller.
[20]	IMSTIFSELERR	Input Data Channel Interface Selection Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:IMSTIFSEL-ERR will be propagated to the interrupt controller.
[19]	OMSTERR	Output Data Channel Master Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:OMSTERR will be propagated to the interrupt controller.

Table 43-10. Interrupt Request Enable Register (SHE_IRQEN) bits

Bit Position	Bit Field Name	Bit Description
[18]	IMSTERR	Input Data Channel Master Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:IMSTERR will be propagated to the interrupt controller.
[17]	OFIFORDERR	Read From Empty Output FIFO Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:OFI-FORDERR will be propagated to the interrupt controller.
[16]	IFIFOWRERR	Write To Full Input FIFO Error Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:IFIFOWRERR will be propagated to the interrupt controller.
[15:8]	read0	-
[7:6]	read0	-
[5]	OFIFORDTH	Output FIFO Above Threshold Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:OFIFORDTH will be propagated to the interrupt controller.
[4]	IFIFOWRTH	Input FIFO Below Threshold Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:IFIFOWRTH will be propagated to the interrupt controller.
[3]	OMSTIDLE	Output Data Channel Master Idle Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:OMSTIDLE will be propagated to the interrupt controller.
[2]	IMSTIDLE	Input Data Channel Master Idle Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:IMSTIDLE will be propagated to the interrupt controller.
[1]	DONE	Command Execution Done Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:DONE will be propagated to the interrupt controller.
[0]	COMPAREMATCH	Compare Match Interrupt Enable bit If set to '1', the interrupt request stored in SHE_IRQ:COM-PAREMATCH will be propagated to the interrupt controller.

43.2.3.3 Interrupt Request Clear Register (SHE_IRQCLR)

This register clears interrupt requests stored in SHE_IRQ. Refer to Interrupt request generation for more information on interrupt generation/processing.

Interrupt Request Clear Register (SHE_IRQCLR)

Figure 43-11. Interrupt Request Clear Register (SHE_IRQCLR)

SHE_IRQCLR																															
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
read0	read0	read0	read0	read0	read0	read0	read0	FATALERR	IFIFOLOCKERR	OMSTIFSELERR	IMSTIFSELERR	OMSTERR	IMSTERR	OFIFORDERR	IFIFOWRERR	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	OFIFORDTH	IFIFOWRTH	OMSTIDLE	IMSTIDLE	DONE	COMPAREMATCH
R0	R0	R0	R0	R0	R0	R0	R0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1	Rp0Wp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-11. Interrupt Request Clear Register (SHE_IRQCLR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23]	FATALERR	Fatal Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:FATALERR.
[22]	IFIFOLOCKERR	Write To Locked Input FIFO Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:IFIFOLOCKERR.
[21]	OMSTIFSELERR	Output Data Channel Interface Selection Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:OMSTIFSELERR.
[20]	IMSTIFSELERR	Input Data Channel Interface Selection Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:IMSTIFSELERR.
[19]	OMSTERR	Output Data Channel Master Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:OMSTERR.

Table 43-11. Interrupt Request Clear Register (SHE_IRQCLR) bits

Bit Position	Bit Field Name	Bit Description
[18]	IMSTERR	Input Data Channel Master Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:IMSTERR.
[17]	OFIFORDERR	Read From Empty Output FIFO Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:OFIFORDERR.
[16]	IFIFOWRERR	Write To Full Input FIFO Error Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:IFIFOWRERR.
[15:8]	read0	-
[7:6]	read0	-
[5]	OFIFORDTH	Output FIFO Above Threshold Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:OFIFORDTH.
[4]	IFIFOWRTH	Input FIFO Below Threshold Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:IFIFOWRTH.
[3]	OMSTIDLE	Output Data Channel Master Idle Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:OMSTIDLE.
[2]	IMSTIDLE	Input Data Channel Master Idle Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:IMSTIDLE.
[1]	DONE	Command Execution Done Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:DONE.
[0]	COMPAREMATCH	Compare Match Interrupt Clear bit Writing this bit to '1' clears the interrupt request stored in SHE_IRQ:COMPAREMATCH.

43.2.4 Configuration registers for the data interface

43.2.4.1 Input Channel Master Start Address Register (SHE_IMSTADDR)

This register contains the memory start address for data transfer by the data interface input channel master. The address stored has to be aligned to the 64-bit boundary.

Moreover, the start address should be aligned to the 128-byte boundary if the transfer is supposed to cross the 4KB address boundary. Otherwise at least one short burst transfer (on reaching the 4KB boundary) will be performed. This limitation comes from the AXI protocol which cannot perform bursts over the 4KB address boundary.

This register must be configured by the user prior to the transfer. After the transfer has started the register is locked for writing. Write access to the locked register produces a bus error response. The lock is released if the required amount of data has been transferred (i.e. SHE_IMSTCNT is zero) or if the data transfer has been canceled.

Input Channel Master Start Address Register (SHE_IMSTADDR)

Figure 43-12. Input Channel Master Start Address Register (SHE_IMSTADDR)

SHE_IMSTADDR																															
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**
IMSTADDR[31]	IMSTADDR[30]	IMSTADDR[29]	IMSTADDR[28]	IMSTADDR[27]	IMSTADDR[26]	IMSTADDR[25]	IMSTADDR[24]	IMSTADDR[23]	IMSTADDR[22]	IMSTADDR[21]	IMSTADDR[20]	IMSTADDR[19]	IMSTADDR[18]	IMSTADDR[17]	IMSTADDR[16]	IMSTADDR[15]	IMSTADDR[14]	IMSTADDR[13]	IMSTADDR[12]	IMSTADDR[11]	IMSTADDR[10]	IMSTADDR[9]	IMSTADDR[8]	IMSTADDR[7]	IMSTADDR[6]	IMSTADDR[5]	IMSTADDR[4]	IMSTADDR[3]	IMSTADDR[2]	IMSTADDR[1]	IMSTADDR[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-12. Input Channel Master Start Address Register (SHE_IMSTADDR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IMSTADDR	Input Channel Master Start Address

43.2.4.2 Output Channel Master Start Address Register (SHE_OMSTADDR)

This register contains the memory start address for data transfer by the data interface output channel master. The address stored has to be aligned to the 64-bit boundary.

Moreover, the start address should be aligned to the 128-byte boundary if the transfer is supposed to cross the 4KB address boundary. Otherwise at least one short burst transfer (on reaching the 4KB boundary) will be performed. This limitation comes from the AXI protocol, which cannot perform bursts over the 4KB address boundary.

This register has to be configured by the user prior to the transfer. After the transfer has started the register is locked for writing. Write access to the locked register produces a bus error response. The lock is released if the required amount of data has been transferred (i.e. SHE_OMSTCNT is zero) or if the data transfer has been canceled.

Output Channel Master Start Address Register (SHE_OMSTADDR)

Figure 43-13. Output Channel Master Start Address Register (SHE_OMSTADDR)

SHE_OMSTADDR																															
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**
OMSTADDR[31]	OMSTADDR[30]	OMSTADDR[29]	OMSTADDR[28]	OMSTADDR[27]	OMSTADDR[26]	OMSTADDR[25]	OMSTADDR[24]	OMSTADDR[23]	OMSTADDR[22]	OMSTADDR[21]	OMSTADDR[20]	OMSTADDR[19]	OMSTADDR[18]	OMSTADDR[17]	OMSTADDR[16]	OMSTADDR[15]	OMSTADDR[14]	OMSTADDR[13]	OMSTADDR[12]	OMSTADDR[11]	OMSTADDR[10]	OMSTADDR[9]	OMSTADDR[8]	OMSTADDR[7]	OMSTADDR[6]	OMSTADDR[5]	OMSTADDR[4]	OMSTADDR[3]	OMSTADDR[2]	OMSTADDR[1]	OMSTADDR[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-13. Output Channel Master Start Address Register (SHE_OMSTADDR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	OMSTADDR	Output Channel Master Start Address

43.2.4.3 Input Channel Master Data Transfer Counter (SHE_IMSTCNT)

This register defines the number of 64-bit double words to be transferred by the input channel master.

It has to be configured by the user prior to starting the transfer. After the transfer has started the register is locked for write accesses and will be updated by the hardware to represent the amount of remaining data transfers. Write access to the locked register produces a bus error response. The lock is released if the required amount of data has been transferred (i.e. SHE_IMSTCNT is zero) or if the data transfer has been canceled.

Input Channel Master Data Transfer Counter (SHE_IMSTCNT)

Figure 43-14. Input Channel Master Data Transfer Counter (SHE_IMSTCNT)

SHE_IMSTCNT																															
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**
read0	read0	read0	IMSTCNT[28]	IMSTCNT[27]	IMSTCNT[26]	IMSTCNT[25]	IMSTCNT[24]	IMSTCNT[23]	IMSTCNT[22]	IMSTCNT[21]	IMSTCNT[20]	IMSTCNT[19]	IMSTCNT[18]	IMSTCNT[17]	IMSTCNT[16]	IMSTCNT[15]	IMSTCNT[14]	IMSTCNT[13]	IMSTCNT[12]	IMSTCNT[11]	IMSTCNT[10]	IMSTCNT[9]	IMSTCNT[8]	IMSTCNT[7]	IMSTCNT[6]	IMSTCNT[5]	IMSTCNT[4]	IMSTCNT[3]	IMSTCNT[2]	IMSTCNT[1]	IMSTCNT[0]
R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-14. Input Channel Master Data Transfer Counter (SHE_IMSTCNT) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:0]	IMSTCNT	Input Channel Master Data Transfer Counter

43.2.4.4 Output Channel Master Data Transfer Counter (SHE_OMSTCNT)

This register defines the number of 64-bit double words to be transferred by the output channel master.

This register has to be configured by the user prior to starting the transfer. After the transfer has started the register is locked for write accesses and will be updated by the hardware to represent the amount of remaining data transfers. Writing access to the locked register produces a bus error response. The lock is released if the required amount of data has been transferred (SHE_OMSTCNT is zero) or if the data transfer has been canceled.

Output Channel Master Data Transfer Counter (SHE_OMSTCNT)

Figure 43-15. Output Channel Master Data Transfer Counter (SHE_OMSTCNT)

SHE_OMSTCNT																															
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**
read0	read0	read0	OMSTCNT[28]	OMSTCNT[27]	OMSTCNT[26]	OMSTCNT[25]	OMSTCNT[24]	OMSTCNT[23]	OMSTCNT[22]	OMSTCNT[21]	OMSTCNT[20]	OMSTCNT[19]	OMSTCNT[18]	OMSTCNT[17]	OMSTCNT[16]	OMSTCNT[15]	OMSTCNT[14]	OMSTCNT[13]	OMSTCNT[12]	OMSTCNT[11]	OMSTCNT[10]	OMSTCNT[9]	OMSTCNT[8]	OMSTCNT[7]	OMSTCNT[6]	OMSTCNT[5]	OMSTCNT[4]	OMSTCNT[3]	OMSTCNT[2]	OMSTCNT[1]	OMSTCNT[0]
R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-15. Output Channel Master Data Transfer Counter (SHE_OMSTCNT) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:0]	OMSTCNT	Output Channel Master Data Transfer Counter

43.2.4.5 Input Channel Master Start Trigger (SHE_IMSTSTART)

This register is used to start data transfer by the input data channel master.

Input Channel Master Start Trigger (SHE_IMSTSTART)

Figure 43-16. Input Channel Master Start Trigger (SHE_IMSTSTART)

SHE_IMSTSTART																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IMSTSTART
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-16. Input Channel Master Start Trigger (SHE_IMSTSTART) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	IMSTSTART	Input Channel Master Start Trigger Writing '1' to this bit starts the transfer of the input channel master. Writing '0' has no effect. The start trigger will be reset through the hardware when the required amount of data has been transferred to the input FIFO or if the data transfer was canceled. If the start of a new data transfer is requested and the input FIFO interface selection is configured to 'Command Interface' (which is in fact an error condition), the SHE_IMSTSTART bit remains zero but the SHE_IRQ:IMSTIFSELERR error interrupt flag is set instead.

43.2.4.6 Output Channel Master Start Trigger (SHE_OMSTSTART)

This register is used to start data transfer by the output data channel master.

Output Channel Master Start Trigger (SHE_OMSTSTART)

Figure 43-17. Output Channel Master Start Trigger (SHE_OMSTSTART)

SHE_OMSTSTART																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	OMSTSTART
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-17. Output Channel Master Start Trigger (SHE_OMSTSTART) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	OMSTSTART	Output Channel Master Start Trigger Writing '1' to this bit starts the transfer of the output channel master. Writing '0' has no effect. The start trigger will be automatically reset through the hardware when the required amount of data has been transferred or if the data transfer was canceled. If the start of new data transfer is requested and the output FIFO interface selection is configured to 'Command Interface' (which is in fact an error condition), the SHE_OMSTSTART bit remains zero but the SHE_IRQ:OMSTIFSELERR error interrupt flag is set instead.

43.2.4.7 Input FIFO Configuration Register (SHE_IFIFOCFG)

This register controls the interface selection as well as the write threshold setting for the input FIFO.

Input FIFO Configuration Register (SHE_IFIFOCFG)

Figure 43-18. Input FIFO Configuration Register (SHE_IFIFOCFG)

SHE_IFIFOCFG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IFSEL	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	WRTHRESHOLD[5]	WRTHRESHOLD[4]	WRTHRESHOLD[3]	WRTHRESHOLD[2]	WRTHRESHOLD[1]	WRTHRESHOLD[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-18. Input FIFO Configuration Register (SHE_IFIFOCFG) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	IFSEL	Interface Selection bit '1': Input FIFO can be written through the Command Interface '0': Input FIFO can be written through the Data Interface If the data transfer has to be performed by the input data channel master, this bit must be configured (to '0') prior to starting the transfer. After the transfer starts, the bit is locked for writing and any attempt to write to it produces a bus error response. The lock is released if the required amount of data has been transferred (i.e. SHE_IMSTCNT is zero) or if the data transfer has been canceled
[15:8]	read0	-
[7:6]	read0	-

Table 43-18. Input FIFO Configuration Register (SHE_IFIFOCFG) bits

Bit Position	Bit Field Name	Bit Description
[5:0]	WRTHRESHOLD	Programmable Write Threshold Of Input FIFO The programmable write threshold of the input FIFO (in terms of 32-bit words) can be used for interrupt-based software design. If the amount of stored 32-bit data words (the amount is defined by the SHE_FIFOLOAD:IFIFOLOAD bits) is less than the value of WRTHRESHOLD, the IFIFOWRTH bit in the SHE_FIFOSTATUS register is set. Moreover, an interrupt request (i.e. SHE_IRQ:IFIFOWRTH) is set on the rising edge of SHE_FIFOSTATUS:IFIFOWRTH. The capacity of the input FIFO is 1,536 bits, i.e. 32 bits wide and 48 words deep. The threshold value is not verified by SHE for its feasibility. This means that if the programmed threshold value exceeds the capacity of the FIFO (i.e. WRTHRESHOLD > 48), the SHE_FIFOSTATUS:IFIFOW- RTH status bit will never be cleared. **Note:** This bit field should be written using 8- or 16-bit accesses in case of pending data transfer by the input data channel master (SHE_MSTSTATUS:IMSTLOCK == '1'). Otherwise there will be a bus error response because the IFSEL bit is locked for writing.

43.2.4.8 Output FIFO Configuration Register (SHE_OFIFOCFG)

This register controls the interface selection as well as the read threshold setting for the output FIFO.

Output FIFO Configuration register (SHE_OFIFOCFG)

Figure 43-19. Output FIFO Configuration Register (SHE_OFIFOCFG)

SHE_OFIFOCFG																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IFSEL	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	RDTHRESHOLD[5]	RDTHRESHOLD[4]	RDTHRESHOLD[3]	RDTHRESHOLD[2]	RDTHRESHOLD[1]	RDTHRESHOLD[0]
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-19. Output FIFO Configuration Register (SHE_OFIFOCFG) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	IFSEL	Interface Selection bit '1': Output FIFO can be read through the Command Interface '0': Output FIFO can be read through the Data Interface If the data transfer has to be performed by the output data channel master, this bit must be configured (to '0') prior to starting the transfer. After the transfer starts, this bit is locked for writing and any attempt to write to this bit produces a bus error response. The lock is released if the required amount of data has been transferred (SHE_OMSTCNT is zero) or if the data transfer has been canceled
[15:8]	read0	-
[7:6]	read0	-

Table 43-19. Output FIFO Configuration Register (SHE_OFIFOCFG) bits

Bit Position	Bit Field Name	Bit Description
[5:0]	RDTHRESHOLD	Programmable Read Threshold Of Output FIFO The programmable read threshold of the output FIFO (in terms of 32- bit words) can be used for interrupt-based software design. If the amount of stored 32-bit data words (the amount is defined by the SHE_FIFOLOAD:OFIFOLOAD bits) is greater than the value of RDTHRESHOLD, the OFIFORDTH bit in the SHE_FIFOSTATUS register is set. Moreover, an interrupt request (SHE_IRQ:OFI-FORDTH) is set on the rising edge of SHE_FIFOSTATUS:OFI-FORDTH. The capacity of the output FIFO is equal to 1,536 bits, i.e. 32 bits wide and 48 words deep. The threshold value is not verified by SHE for its feasibility. This means that if the programmed threshold value exceeds the capacity of the FIFO (i.e. RDTHRESHOLD > 47), the SHE_FIFOSTATUS:OFIFORDTH status bit will never be set. **Note:** This bit field should be written using 8- or 16-bit accesses in case of pending data transfer by the input data channel master (SHE_MSTSTATUS:OMSTLOCK == '1'). Other-wise there will be a bus error response because the IFSEL bit is locked for writing.

43.2.4.9 Input FIFO Compare Value Register (SHE_COMPARE0~1)

This register contains the amount of data (all parameters and the actual data used for processing) required for a single SHE API command represented in terms of 32-bit data words.

SHE_COMPARE is used by the hardware to recognize that the required amount of data for a single SHE command has been transferred through the input FIFO. For this purpose the value of the compare register SHE_COMPARE is continuously compared with the value of the data counter register SHE_DATACNT. When the data counter reaches the value of the compare register, the status bit SHE_FIFOSTATUS:COMPAREMATCH and an interrupt request bit SHE_IRQ:COMPAREMATCH are immediately set. This transition of the COMPAREMATCH bit is referenced to as a compare match event.

SHE_COMPARE is reset by the hardware to the maximal value (2^{59}-1) every time the opcode for a new SHE command is written to the SHE_CMD register.

In most cases the SHE_COMPARE register is configured by the control logic of SHE according to selected command and/or parameter values and cannot be changed during command execution. But for CBC commands, the software can adjust the value of SHE_COMPARE during the command execution, i. e. if the total amount of data is not known before the command start. Since the SHE_COMPARE register is 59 bits wide and accessed using a 32-bit interface, it cannot be written through an atomic operation and a faulty compare match event can occur. Hence the software has to use the following approach, when adjusting it: To increase the compare value, the software can write the most significant 32 bits first and then adjust the least significant 32 bits. To decrease the compare value, the software can write the least significant 32 bits first and then adjust the most significant 32 bits.

Note: The software is only all owed to change the value of the SHE_COMPARE register if a CBC command is performed. Otherwise a bus error response is generated.

Input FIFO Compare Value Register (SHE_COMPARE0~1)

Figure 43-20. Input FIFO Compare Value Register 0 (SHE_COMPARE0)

SHE_COMPARE0																															
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**
COMPARE[31]	COMPARE[30]	COMPARE[29]	COMPARE[28]	COMPARE[27]	COMPARE[26]	COMPARE[25]	COMPARE[24]	COMPARE[23]	COMPARE[22]	COMPARE[21]	COMPARE[20]	COMPARE[19]	COMPARE[18]	COMPARE[17]	COMPARE[16]	COMPARE[15]	COMPARE[14]	COMPARE[13]	COMPARE[12]	COMPARE[11]	COMPARE[10]	COMPARE[9]	COMPARE[8]	COMPARE[7]	COMPARE[6]	COMPARE[5]	COMPARE[4]	COMPARE[3]	COMPARE[2]	COMPARE[1]	COMPARE[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 43-20. Input FIFO Compare Value Register 0 (SHE_COMPARE0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	COMPARE	Least significant bits of the compare register The amount of data required for a single SHE Application Programming Interface (API) command

Figure 43-21. Input FIFO Compare Value Register 1 (SHE_COMPARE1)

SHE_COMPARE1																															
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**
read0	read0	read0	read0	read0	COMPARE[26]	COMPARE[25]	COMPARE[24]	COMPARE[23]	COMPARE[22]	COMPARE[21]	COMPARE[20]	COMPARE[19]	COMPARE[18]	COMPARE[17]	COMPARE[16]	COMPARE[15]	COMPARE[14]	COMPARE[13]	COMPARE[12]	COMPARE[11]	COMPARE[10]	COMPARE[9]	COMPARE[8]	COMPARE[7]	COMPARE[6]	COMPARE[5]	COMPARE[4]	COMPARE[3]	COMPARE[2]	COMPARE[1]	COMPARE[0]
R0	R0	R0	R0	R0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 43-21. Input FIFO Compare Value Register 1 (SHE_COMPARE1) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26:0]	COMPARE	Most significant bits of the compare register The amount of data required for a single SHE API command

43.2.5 Status registers for data interface

43.2.5.1 Compare Register Access Status Register (SHE_COMPACC)

This register represents the current status of the write access permissions to the SHE_COMPARE register.

Compare Register Access Status Register (SHE_COMPACC)

Figure 43-22. Compare Register Access Status Register (SHE_COMPACC)

SHE_COMPACC																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CPUEN
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-22. Compare Register Access Status Register (SHE_COMPACC) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7:1]	read0	-
[0]	CPUEN	CPU Write Access Enabled status flag '1': The software is allowed to write to the SHE_COMPARE register '0': The software is not allowed to write to the SHE_COMPARE register

43.2.5.2 Data Master Status Register (SHE_MSTSTATUS)

This register represents the current status of the input and output channel data masters.

Data Master Status Register (SHE_MSTSTATUS)

Figure 43-23. Data Master Status Register (SHE_MSTSTATUS)

SHE_MSTSTATUS																															
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**
read0	read0	read0	read0	read0	OMSTERRRESP[1]	OMSTERRRESP[0]	OMSTERR	read0	read0	read0	read0	read0	read0	OMSTLOCK	OMSTIDLE	read0	read0	read0	read0	read0	IMSTERRRESP[1]	IMSTERRRESP[0]	IMSTERR	read0	read0	read0	read0	read0	read0	IMSTLOCK	IMSTIDLE
R0	R0	R0	R0	R0	Rp	Rp	Rp	R0	R0	R0	R0	R0	R0	Rp	Rp	R0	R0	R0	R0	R0	Rp	Rp	Rp	R0	R0	R0	R0	R0	R0	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 43-23. Data master status register (SHE_MSTSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26:25]	OMSTERRRESP	Output Data Channel Master Error Response Code These bits contain the error response code of the AXI protocol. '00': OKAY (Normal access okay) A normal access has been successful. This code can also indicate an exclusive access failure '01': EXOKAY (Exclusive access okay) Either the read or write portion of an exclusive access has been successful '10': SLVERR (Slave error) Access has reached the slave successfully but the slave wishes to return an error condition to the originating master '11': DECERR (Decode error) Typically, a decode error is generated by an interconnect component to indicate that there is no slave at the transaction address The register is updated in the event of an AXI bus error response on the output data channel.

Table 43-23. Data master status register (SHE_MSTSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[24]	OMSTERR	Output Data Channel Master Error Response flag This bit is set in the event of an AXI bus error response on the output data channel. It is cleared on the rising edge of the DONE flag (SHE_STATUS:DONE). The rising edge of the OMSTERR error flag sets the SHE_IRQ:OMSTERR interrupt flag.
[23:18]	read0	-
[17]	OMSTLOCK	Output Data Channel Master Lock Enabled flag If this bit is high, the output master channel configuration registers (SHE_OFIFOCFG:IFSEL, SHE_OMSTADDR and SHE_OMSTCNT) are locked for writing. The Output Data Channel Master Lock Enabled flag is set at the start of a data transfer. And it is cleared after the required amount of data has been transferred (i.e. SHE_OMSTCNT is zero), or if the data transfer was canceled and the current burst finished.
[16]	OMSTIDLE	Output Data Channel Master Idle flag This bit is high if no data transfer is ongoing. It is cleared at the start of a new data transfer. The Output Data Channel Master Idle flag is set when data transfer has completed, i.e. the Output Channel Master Data Transfer Counter (SHE_OMSTCNT) is zero and all the data has been transferred to the requested memory location, or if data transfer has been canceled and the current burst finished. The rising edge of the OMSTIDLE flag sets SHE_IRQ:OMSTIDLE interrupt flag.
[15:11]	read0	-

Table 43-23. Data master status register (SHE_MSTSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[10:9]	IMSTERRRESP	Input Data Channel Master Error Response Code These bits contain the error response code of the AXI protocol. '00': OKAY (Normal access okay) Normal access has been successful. This value can also indicate an exclusive access failure '01': EXOKAY (Exclusive access okay) Either the read or write portion of an exclusive access has been successful '10': SLVERR (Slave error) Slave error is used when the access has successfully reached the slave but the slave wishes to return an error condition to the originating master '11': DECERR (Decode error) Typically, a decode error is generated by an interconnect component to indicate that there is no slave at the transaction address The register is updated in the event of an AXI bus error response on the input data channel.
[8]	IMSTERR	Input Data Channel Master Error Response flag This bit is set in the event of an AXI bus error response on the input data channel. It is cleared on the rising edge of the DONE flag (SHE_STATUS:DONE). The rising edge of IMSTERR error flag sets the SHE_IRQ:IMSTERR interrupt flag.
[7:2]	read0	-
[1]	IMSTLOCK	Input Data Channel Master Lock Enabled flag If this bit is high, the input master channel configuration registers (SHE_IFIFOCFG:IFSEL, SHE_IMSTADDR and SHE_IMSTCNT) are locked for writing. It is set at the start of a data transfer. The Input Data Channel Master Lock Enabled flag is cleared after the required amount of data has been transferred (i.e. SHE_IMSTCNT is zero), or if the data transfer was canceled and current burst has been finished.

Table 43-23. Data master status register (SHE_MSTSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[0]	IMSTIDLE	Input Data Channel Master Idle flag This bit is high if no data transfer is ongoing. It is cleared at the start of a new data transfer. The Input Data Channel Master Idle flag is set when the data transfer to the input FIFO has completed, i.e. the Input Channel Master Data Transfer Counter (SHE_IMSTCNT) is zero, or if the data transfer has been canceled and the current burst finished. The rising edge of the IMSTIDLE flag sets the SHE_IRQ:IMSTIDLE interrupt flag.

43.2.5.3 Input Channel Master Error Response Address Register (SHE_IMSTERRADDR)

This register is updated in case of an AXI bus error response on the input data channel and contains the address of the slave which issued the error response. The register content is valid until the next error response occurs.

In a read transaction, the AXI slave can give different responses for different transfers within a burst. Hence the SHE_IMSTERRADDR register stores the precise address.

Input Channel Master Error Response Address Register (SHE_IMSTERRADDR)

Figure 43-24. Input Channel Master Error Response Address Register (SHE_IMSTERRADDR)

SHE_IMSTERRADDR																															
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**
IMSTERRADDR[31]	IMSTERRADDR[30]	IMSTERRADDR[29]	IMSTERRADDR[28]	IMSTERRADDR[27]	IMSTERRADDR[26]	IMSTERRADDR[25]	IMSTERRADDR[24]	IMSTERRADDR[23]	IMSTERRADDR[22]	IMSTERRADDR[21]	IMSTERRADDR[20]	IMSTERRADDR[19]	IMSTERRADDR[18]	IMSTERRADDR[17]	IMSTERRADDR[16]	IMSTERRADDR[15]	IMSTERRADDR[14]	IMSTERRADDR[13]	IMSTERRADDR[12]	IMSTERRADDR[11]	IMSTERRADDR[10]	IMSTERRADDR[9]	IMSTERRADDR[8]	IMSTERRADDR[7]	IMSTERRADDR[6]	IMSTERRADDR[5]	IMSTERRADDR[4]	IMSTERRADDR[3]	IMSTERRADDR[2]	IMSTERRADDR[1]	IMSTERRADDR[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-24. Input Channel Master Error Response Address Register (SHE_IMSTERRADDR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IMSTERRADDR	Input Data Channel Error Response Address

43.2.5.4 Output Channel Master Error Response Address Register (SHE_OMSTERRADDR)

This register is updated in the event of the AXI bus error response on the output data channel and contains the address of the slave which issued the error response. The register content is valid until the next error response occurs.

For a write transaction, there is just one response given for the entire burst and not for each data transfer within the burst. Hence the SHE_OMSTERRADDR register stores the imprecise address (i.e. the start address of the burst).

Output Channel Master Error Response Address Register (SHE_OMSTERRADDR)

Figure 43-25. Output Channel Master Error Response Address Register (SHE_OMSTERRADDR)

SHE_OMSTERRADDR																															
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**
OMSTERRADDR[31]	OMSTERRADDR[30]	OMSTERRADDR[29]	OMSTERRADDR[28]	OMSTERRADDR[27]	OMSTERRADDR[26]	OMSTERRADDR[25]	OMSTERRADDR[24]	OMSTERRADDR[23]	OMSTERRADDR[22]	OMSTERRADDR[21]	OMSTERRADDR[20]	OMSTERRADDR[19]	OMSTERRADDR[18]	OMSTERRADDR[17]	OMSTERRADDR[16]	OMSTERRADDR[15]	OMSTERRADDR[14]	OMSTERRADDR[13]	OMSTERRADDR[12]	OMSTERRADDR[11]	OMSTERRADDR[10]	OMSTERRADDR[9]	OMSTERRADDR[8]	OMSTERRADDR[7]	OMSTERRADDR[6]	OMSTERRADDR[5]	OMSTERRADDR[4]	OMSTERRADDR[3]	OMSTERRADDR[2]	OMSTERRADDR[1]	OMSTERRADDR[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-25. Output Channel Master Error Response Address Register (SHE_OMSTERRADDR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	OMSTERRADDR	Output Data Channel Error Response Address

43.2.5.5 FIFO Status Register (SHE_FIFOSTATUS)

This register represents the current status of the input and output FIFOs.

FIFO Status Register (SHE_FIFOSTATUS)

Figure 43-26. FIFO Status Register (SHE_FIFOSTATUS)

SHE_FIFOSTATUS																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	COMPAREMATCH	read0	read0	read0	read0	read0	read0	read0	OFIFORDTH	read0	read0	read0	read0	read0	read0	read0	IFIFOWRTH
R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	R0	Rp	R0	R0	R0	R0	R0	R0	R0	Rp	R0	R0	R0	R0	R0	R0	R0	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-26. FIFO Status Register (SHE_FIFOSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	read0	-
[23:17]	read0	-
[16]	COMPAREMATCH	Compare Match Event flag This bit signalises that the required amount of data has been transferred through the input FIFO and its write port is locked. It is set automatically if the value of the SHE_DATACNT register is greater than or equal to the value of the SHE_COMPARE register and remains high until the start of the next command. Increasing the value of the SHE_COMPARE register above the value of the SHE_DATACNT register does not clear the asserted COMPAREMATCH flag.
[15:9]	read0	-
[8]	OFIFORDTH	Output FIFO Above Threshold flag This bit is set if the amount of data words stored in the output FIFO (SHE_FIFOLOAD:OFIFOLOAD) is greater than the value configured in the read threshold register (SHE_OFIFOCFG:RDTHRESHOLD). Otherwise this bit is cleared.
[7:1]	read0	-

Table 43-26. FIFO Status Register (SHE_FIFOSTATUS) bits

Bit Position	Bit Field Name	Bit Description
[0]	IFIFOWRTH	Input FIFO Below Threshold flag This bit is set if the amount of data words stored in the input FIFO (SHE_FIFOLOAD:IFIFOLOAD) is less than the value configured in the write threshold register (SHE_IFIFOCFG:WRTHRESHOLD). Otherwise this bit is cleared.

43.2.5.6 FIFO Load Register (SHE_FIFOLOAD)

This register represents the current load status of the input and output FIFOs.

FIFO Load Register (SHE_FIFOLOAD)

Figure 43-27. FIFO Load Register (SHE_FIFOLOAD)

SHE_FIFOLOAD																															
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**
read0	read0	OFIFOLOAD[5]	OFIFOLOAD[4]	OFIFOLOAD[3]	OFIFOLOAD[2]	OFIFOLOAD[1]	OFIFOLOAD[0]	read0	read0	OFIFOFREE[5]	OFIFOFREE[4]	OFIFOFREE[3]	OFIFOFREE[2]	OFIFOFREE[1]	OFIFOFREE[0]	read0	read0	IFIFOLOAD[5]	IFIFOLOAD[4]	IFIFOLOAD[3]	IFIFOLOAD[2]	IFIFOLOAD[1]	IFIFOLOAD[0]	read0	read0	IFIFOFREE[5]	IFIFOFREE[4]	IFIFOFREE[3]	IFIFOFREE[2]	IFIFOFREE[1]	IFIFOFREE[0]
R0	R0	Rp	Rp	Rp	Rp	Rp	Rp	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 43-27. FIFO Load Register (SHE_FIFOLOAD) bits

Bit Position	Bit Field Name	Bit Description
[31:30]	read0	-
[29:24]	OFIFOLOAD	The amount of data stored in the output FIFO is defined in terms of 32- bit data words
[23:22]	read0	-
[21:16]	OFIFOFREE	The amount of data which can be written into the output FIFO is defined in terms of 32-bit data words
[15:14]	read0	-
[13:8]	IFIFOLOAD	The amount of data stored in the input FIFO is defined in terms of 32- bit data words
[7:6]	read0	-
[5:0]	IFIFOFREE	The amount of data which can be written into the input FIFO is defined in terms of 32-bit data words

43.2.5.7 Input FIFO Data Counter Register (SHE_DATACNT0~1)

This register represents the amount of 32-bit data words transferred through the input FIFO.

SHE_DATACNT is used by the hardware to recognize that the required amount of data for a single SHE command has been transferred through the input FIFO. For this purpose the value of the compare register SHE_COMPARE is continuously compared with the value of the data counter register SHE_DATACNT. When the data counter reaches the value of the compare register, the status bit SHE_FIFOSTATUS:COMPAREMATCH and an interrupt request bit SHE_IRQ:COMPAREMATCH are immediately set. This transition of the COMPAREMATCH bit is referenced to as a compare match event.

SHE_DATACNT is reset by the hardware to zero every time the opcode for the new SHE command is written to the SHE_CMD register.

Input FIFO Data Counter Register (SHE_DATACNT0~1)

Figure 43-28. Input FIFO Data Counter Register 0 (SHE_DATACNT0)

SHE_DATACNT0																															
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**
DATACNT[31]	DATACNT[30]	DATACNT[29]	DATACNT[28]	DATACNT[27]	DATACNT[26]	DATACNT[25]	DATACNT[24]	DATACNT[23]	DATACNT[22]	DATACNT[21]	DATACNT[20]	DATACNT[19]	DATACNT[18]	DATACNT[17]	DATACNT[16]	DATACNT[15]	DATACNT[14]	DATACNT[13]	DATACNT[12]	DATACNT[11]	DATACNT[10]	DATACNT[9]	DATACNT[8]	DATACNT[7]	DATACNT[6]	DATACNT[5]	DATACNT[4]	DATACNT[3]	DATACNT[2]	DATACNT[1]	DATACNT[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-28. Input FIFO Data Counter Register 0 (SHE_DATACNT0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	DATACNT	Least significant bits of the data counter register The amount of data which has been transferred through the input FIFO is defined in terms of 32-bit data words.

Figure 43-29. Input FIFO Data Counter Register 1 (SHE_DATACNT1)

SHE_DATACNT1																															
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**
read0	read0	read0	read0	read0	DATACNT[26]	DATACNT[25]	DATACNT[24]	DATACNT[23]	DATACNT[22]	DATACNT[21]	DATACNT[20]	DATACNT[19]	DATACNT[18]	DATACNT[17]	DATACNT[16]	DATACNT[15]	DATACNT[14]	DATACNT[13]	DATACNT[12]	DATACNT[11]	DATACNT[10]	DATACNT[9]	DATACNT[8]	DATACNT[7]	DATACNT[6]	DATACNT[5]	DATACNT[4]	DATACNT[3]	DATACNT[2]	DATACNT[1]	DATACNT[0]
R0	R0	R0	R0	R0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-29. Input FIFO Data Counter Register 0 (SHE_DATACNT0) bits

Bit Position	Bit Field Name	Bit Description
[31:27]	read0	-
[26:0]	DATACNT	Most significant bits of the data counter register The amount of data which has been transferred through the input FIFO is defined in terms of 32-bit data words.

43.2.6 Data transfer registers

43.2.6.1 Input FIFO Write Data Register (SHE_IFIFOWRDATA0~31)

The input FIFO write data registers are used to write data to the input FIFO through the command interface. A write access to any SHE_IFIFOWRDATA0~31 register is directly mapped to the write port of the input FIFO.

For this reason effectively there is only one register which is replicated thirty-two times in the register map. This is done in order to increase performance of data transfers to the input FIFO by allowing burst transfers over the AHB bus (command interface of SHE). In other words, if several data words have to be transferred into the input FIFO of SHE through the command interface, this should be performed using burst transfer feature of the AHB bus, which is more efficient than several single transfers.

Only 32-bit accesses are allowed to these registers. 64-bit accesses are split into two 32- bit accesses. 8-bit or 16-bit accesses to these register trigger an error response.

The software should check the load status of the FIFO before writing to it. In case the FIFO is full, an error interrupt flag (SHE_IRQ:IFIFOWRERR) is set and the data is lost. Similarly, an error interrupt flag (SHE_IRQ:IFIFOLOCKERR) is set and the data is lost if a write access to the locked input FIFO is performed. An error response on the command interface (AHB bus) is generated in case of a write access to this register if the data interface is selected through the SHE_IFIFOCFG:IFSEL bit.

Input FIFO Write Data Register 0 (SHE_IFIFOWRDATA0)

Figure 43-30. Input FIFO Write Data Register 0 (SHE_IFIFOWRDATA0)

SHE_IFIFOWRDATA0																															
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**
IFIFOWRDATA[31]	IFIFOWRDATA[30]	IFIFOWRDATA[29]	IFIFOWRDATA[28]	IFIFOWRDATA[27]	IFIFOWRDATA[26]	IFIFOWRDATA[25]	IFIFOWRDATA[24]	IFIFOWRDATA[23]	IFIFOWRDATA[22]	IFIFOWRDATA[21]	IFIFOWRDATA[20]	IFIFOWRDATA[19]	IFIFOWRDATA[18]	IFIFOWRDATA[17]	IFIFOWRDATA[16]	IFIFOWRDATA[15]	IFIFOWRDATA[14]	IFIFOWRDATA[13]	IFIFOWRDATA[12]	IFIFOWRDATA[11]	IFIFOWRDATA[10]	IFIFOWRDATA[9]	IFIFOWRDATA[8]	IFIFOWRDATA[7]	IFIFOWRDATA[6]	IFIFOWRDATA[5]	IFIFOWRDATA[4]	IFIFOWRDATA[3]	IFIFOWRDATA[2]	IFIFOWRDATA[1]	IFIFOWRDATA[0]
Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp	Rp0Wp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-30. Input FIFO Write Data Register 0 (SHE_IFIFOWRDATA0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	IFIFOWRDATA	A 32-bit register for accessing the write port of the input FIFO over the Command Interface

43.2.6.2 Output FIFO Read Data Register (SHE_OFIFORDDATA0~31)

The output FIFO read data registers are used to read data from the output FIFO through the command interface. A read access to any SHE_OFIFORDDATA0~31 register is directly mapped to the read port of the output FIFO.

For this reason effectively there is only one register which is replicated thirty-two times in the register map. This is done in order to increase performance of data transfers from the output FIFO by allowing burst transfers over the AHB bus (command interface of SHE). In other words, if several data words have to be transferred from the output FIFO of SHE through the command interface, this should be performed using the burst transfer feature of the AHB bus, which is more efficient than several single transfers.

Only 32-bit accesses are allowed to these registers. 64-bit accesses are split in two 32-bit accesses. 8-bit or 16-bit accesses to these register trigger an error response.

The software should check the load state of the FIFO before reading from it. In case the FIFO is empty, an error interrupt flag (SHE_IRQ:OFIFORDERR) is set and default data is read. An error response is generated in case of read access to this register if data interface is selected through SHE_OFIFOCFG:IFSEL bit.

Output FIFO Read Data Register 0 (SHE_OFIFORDDATA0)

Figure 43-31. Output FIFO Read Data Register 0 (SHE_OFIFORDDATA0)

SHE_OFIFORDDATA0																															
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**
OFIFORDDATA[31]	OFIFORDDATA[30]	OFIFORDDATA[29]	OFIFORDDATA[28]	OFIFORDDATA[27]	OFIFORDDATA[26]	OFIFORDDATA[25]	OFIFORDDATA[24]	OFIFORDDATA[23]	OFIFORDDATA[22]	OFIFORDDATA[21]	OFIFORDDATA[20]	OFIFORDDATA[19]	OFIFORDDATA[18]	OFIFORDDATA[17]	OFIFORDDATA[16]	OFIFORDDATA[15]	OFIFORDDATA[14]	OFIFORDDATA[13]	OFIFORDDATA[12]	OFIFORDDATA[11]	OFIFORDDATA[10]	OFIFORDDATA[9]	OFIFORDDATA[8]	OFIFORDDATA[7]	OFIFORDDATA[6]	OFIFORDDATA[5]	OFIFORDDATA[4]	OFIFORDDATA[3]	OFIFORDDATA[2]	OFIFORDDATA[1]	OFIFORDDATA[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43-31. Output FIFO Read Data Register 0 (SHE_OFIFORDDATA0) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	OFIFORDDATA	A 32-bit register for accessing the read port of the output FIFO over the Command Interface

43.3 Operation of the Secure Hardware Extension

This chapter describes the operation of the SHE module in detail.

43.3.1 Operation of the command interface

Command interface of SHE is mainly used for acessing configuration and status registers of SHE. Additionally it can also be used for the transfer of parameters and data that are used to process commands defined in the SHE Functional Specification.

Feature List

Features of the command interface are:

- Provides configuration and status registers for SHE
- 32-bit Advanced High-Performance Bus (AHB) slave interface
- Command register locked during command execution
- There is a dedicated register for starting the CMD_CANCEL
- FIFO based transfer of parameters, processing data and results of command execution
- Support for interrupt-based software design
- Access protection through Peripheral Protection Unit (PPU) and master ID filter

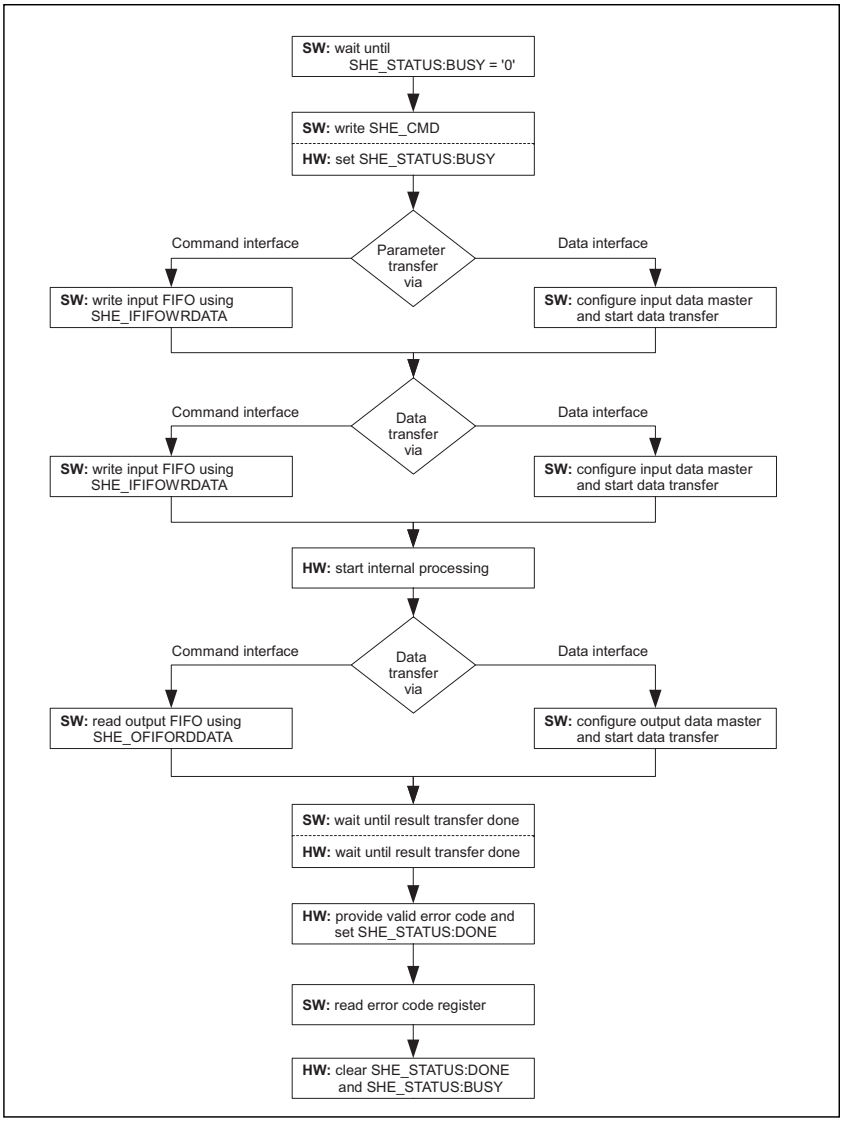
Command execution

The following section describes the execution of SHE API commands inside the SHE module. All commands are defined in the SHE Functional Specification (7. Memory Protection Unit for AHB) and can be separated into two groups: sequential and concurrent. As the name implies, sequential commands can only be executed sequentially, i.e. one command must finish before the next one starts. Concurrent commands can be executed in parallel to other commands, i.e. they can be started at an arbitrary point in time.

Sequential command execution

The processing of a sequential SHE command is presented in Figure 43-32 and is described in the following.

Figure 43-32. Flow chart for the execution of a sequential SHE command



A sequential SHE command can only be started if SHE is in idle mode, i.e. the BUSY bit in the SHE status register (SHE_STATUS) is low. In order to start the execution of the required SHE command, the user has to write the command opcode into the command register (see 43.2.1.1 SHE Command Register (SHE_CMD) description). The commands are defined in the SHE Functional Specification (7. Memory Protection Unit for AHB) and listed in Table 43-32 below.

Note:

The concurrent commands CMD_CANCEL and CMD_GET_STATUS are not present in the overview table. CMD_CANCEL has its own command register (SHE_CMDCANCEL). CMD_GET_STATUS just performs a read access to the status register of SHE (SHE_STATUS) and therefore doesn't need to write an opcode to the command register.

Table 43-32. Sequential SHE commands started through the SHE_CMD register

CMD_ENCRYPT_ECB
CMD_ENCRYPT_CBC
CMD_DECRYPT_ECB
CMD_DECRYPT_CBC

Table 43-32. Sequential SHE commands started through the SHE_CMD register

CMD_GENERATE_MAC
CMD_VERIFY_MAC
CMD_LOAD_KEY
CMD_LOAD_PLAIN_KEY
CMD_EXPORT_RAM_KEY
CMD_INIT_RNG
CMD_EXTEND_SEED
CMD_RND
CMD_SECURE_BOOT
CMD_BOOT_FAILURE
CMD_BOOT_OK
CMD_GET_ID
CMD_DEBUG

Refer to the description of the SHE Command Register (SHE_CMD) register for the opcode values.

As already mentioned, writing command opcode to the SHE_CMD register starts command execution. The busy flag (SHE_STATUS:BUSY) is set and the command register lock is enabled thereupon. A locked command register disables write access to it and protects it from being overwritten by accident or by malicious software. This lock is kept enabled until the complete processing has finished, i.e. as long as the busy flag (SHE_STATUS:BUSY) is high. While the command register is locked only read accesses are allowed. Any write access will trigger a bus error response.

After starting the command execution, the user has to provide the parameters and data which are required for the processing using the input FIFO of SHE. The input FIFO can be written either through the command interface (see 43.2.6.1 Input FIFO Write Data Register (SHE_IFIFOWRDATA0~31) - SHE_IFIFOWRDATA register) or through the SHE data interface. Data transfer through the data interface is done using the input data master, which is capable of performing autonomous memory accesses.

The decision concerning which interface to use should be based on the amount of parameters and data which are required for processing. If only a few data transfers are required, it is more reasonable to use the SHE command interface because of the overhead due to the configuration of the input data master. If SHE is utilized for the processing of bigger data streams, e.g. CBC or Media Access Control (MAC) commands, the data interface input data master should be taken and configured accordingly. Refer to the description of the data interface for more information.

Under normal circumstances both FIFOs are empty after command execution. But there are some error scenarios that may lead to inconsistent FIFO states (e.g. more data than required was fed to the input FIFO). Hence, in order to avoid incorrect behavior from SHE or faulty computation results if some data is still present in the FIFOs, both FIFOs are cleared at the start a new command.

SHE expects the delivery of parameters and data for processing in a pre-defined order. This order as well as the order in which results are generated is shown in Table 43-33. The following commands are not presented in the table since they neither require any parameters or data for processing nor generate results:

- CMD_INIT_RNG
- CMD_BOOT_FAILURE
- CMD_BOOT_OK
- CMD_CANCEL
- CMD_GET_STATUS

In Table 43-33 "Sequential order for parameter and result transfer", please note the reversed parameter and result order for the CMD_DEBUG command: the CHALLENGE value is first provided by SHE and the AUTHORIZATION value is expected afterwards.

Each parameter has to be provided in one or several 64-bit large data blocks. Therefore, parameters with lengths less than 64 bits have to be padded with leading zeros and those wider than 64 bits have to be split into 64-bit large data blocks. The least significant part of the parameter must be transferred first. The same approach is also valid for SHE command results: small results are padded and large results are split.

Table 43-33. Sequential order for parameter and result transfer

Command	Parameter			Return Value		
	Seq. No.	Name	Size [Bits]	Seq. No.	Name	Size [Bits]
CMD_ENC_ECB	1	KEY_ID	64			
	2	PLAINTEXT	128			
				1	CIPHERTEXT	128
CMD_ENC_CBC	1	KEY_ID	64			
	2	IV	128			
	3	PLAINTEXT	n x 128	1	CIPHERTEXT	n x 128
CMD_DEC_ECB	1	KEY_ID	64			
	2	CIPHERTEXT	128			
				1	PLAINTEXT	128
CMD_DEC_CBC	1	KEY_ID	64			
	2	IV	128			
	3	CIPHERTEXT	n x 128			
				1	PLAINTEXT	n x 128
CMD_GENERATE_MAC	1	KEY_ID	64	1	MAC	128
	2	MESSAGE_LENGTH	64			
	3	MESSAGE	n_1 x 128			
CMD_VERIFY_MAC	1	KEY_ID	64			
	2	MESSAGE_LENGTH	64			
	3	MAC_LENGTH	64			
	4	MAC	128			
	5	MESSAGE	n_1 x 128			
				1	VERIFICATION_STATUS	64
CMD_LOAD_KEY	1	M1	128			
	2	M2	256			
	3	M3	128			
				1	M4	256
				2	M5	128
CMD_- LOAD_PLAIN_KEY	1	KEY	128		N/A	N/A
CMD_EXPORT_ RAM_KEY		N/A	N/A	1	M1	128
				2	M2	256
				3	M3	128
				4	M4	256
				5	M5	128

Table 43-33. Sequential order for parameter and result transfer

Command	Parameter			Return Value		
	Seq. No.	Name	Size [Bits]	Seq. No.	Name	Size [Bits]
CMD_EXTEND_SEED	1	ENTROPY	128		N/A	N/A
CMD_RND		N/A	N/A	1	RND	128
CMD_SECURE_-BOOT[2]	1 2	SIZE DATA	64 n_3 x 128		N/A	N/A
CMD_GET_ID	1	CHALLENGE	128	1 2	ID |[4] SREG MAC	120 + 8 128
CMD_DEBUG	1	AUTHORIZATION	128	1	CHALLENGE	128

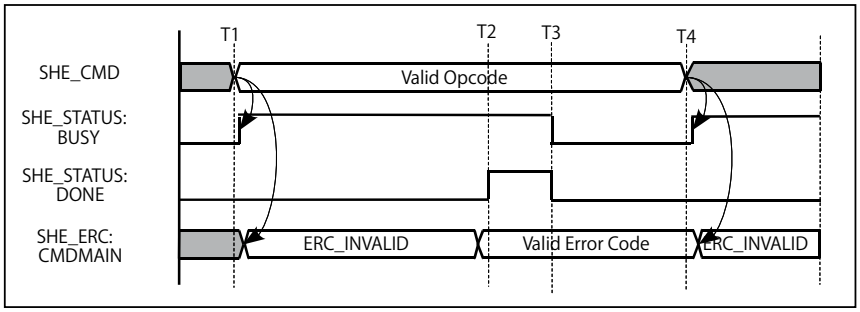
[1] : n = ceil[5] (MESSAGE_LENGTH / 128)
[2] : CMD_SECURE_BOOT is always started by the Boot ROM of the host MCU and is not directly available for the user
[3] : n = ceil[5] (SIZE / 16)
[4] : Vertical bar ('|') symbol represents concatenation of two values. Left value is kept in most significant, right value in least significant bits.
[5] : ceil(x) = min{n ∈ $\aleph_1$ | n ≥ x}, $\aleph_1$ = {1,2,3,...}

The processed results from SHE are written to the output FIFO in the order given in Table 43-33 above and can be read by the user either through the SHE command interface - see the Output FIFO Read Data Register (SHE_OFIFORDDATA0~31) - or data interface using the output data master. Similar to the input data master, the output data master should be used if many data transfers are required.

Once the complete result data has been read from the output FIFO, the SHE control logic writes valid error code to the error code register (SHE_ERC:CMDMAIN) and indicates the end of processing by setting the DONE status flag in the status register (SHE_STATUS). For the commands which require no parameters or do not generate results, as well as for the CMD_DEBUG command, the error code register (SHE_ERC:CMDMAIN) is updated and the SHE_STATUS:DONE flag is set once internal processing has completed.

Command execution is finished only when the software reads the error code register using a 32-bit data access. The SHE_STATUS:BUSY and SHE_STATUS:DONE flags are cleared thereupon. Figure 43-33 illustrates the timing flow for the execution of a sequential SHE command. The command is started at moment T1 by a write access to the SHE_CMD register. Internal processing is finished at moment T2 and the error code register is read at moment T3. A new command can be started at any time after T3, for example at moment T4.

Figure 43-33. Timing diagram for execution of a sequential SHE command



Special case handling for CBC operations

The majority of SHE commands have a fixed number of parameters, and the amount of data used for processing is also fixed. Hence SHE control logic knows the required amount of information which has to be transferred through the input FIFO. This value is stored in a special compare register - see Input FIFO Compare Value Register (SHE_COMPARE0~1) for more details - which is configured through the SHE control logic upon the start of a command.

For two CBC commands, CMD_ENC_CBC and CMD_DEC_CBC, the amount of data used for processing is not always known before command execution (i.e. a priori). Hence command execution is started using the default value of SHE_COMPARE. Once the required amount of data is known, the software has to update the value of the SHE_COMPARE register. For example if SHE is used for a CBC command on a 1024-bit data stream, the total amount of data is 1,216 bits (64 bits for the key ID, 128 bits for initialization vector and 1,024 bits for the data itself). And since SHE_COMPARE represents data in terms of 32-bit data words, it has to be set by the software to 38 (1216 divided by 32). Finally, the software has two possibilities of finishing an ongoing CBC command: adjusting the value of the SHE_COMPARE register; or issuing the CMD_CANCEL command. If CMD_CANCEL is used, the software has to make sure that the required amount of data has been processed in SHE and the results transferred from the SHE output FIFO.

Note: The software is only allowed to change the value of SHE_COMPARE if a CBC command is performed. Otherwise a bus error response is generated.

Concurrent command execution

CMD_GET_STATUS

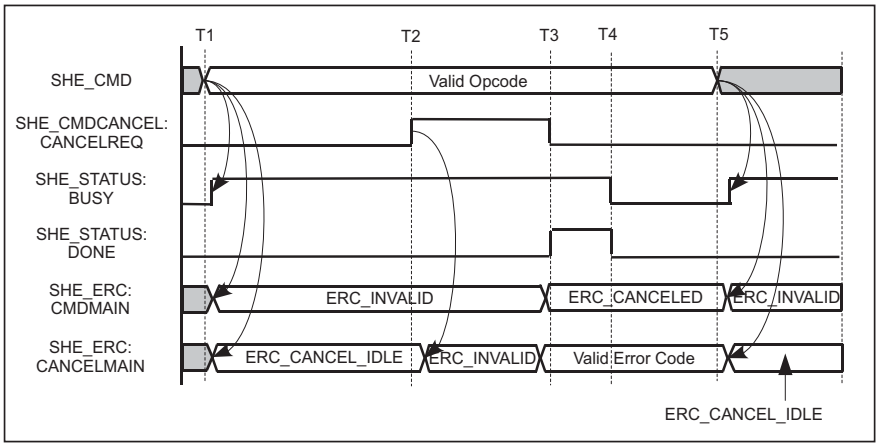
The current status of SHE is provided in SHE_STATUS register which is always kept up to date by the SHE control logic. Since this register is directly accessible by the software, no special handling of the CMD_GET_STATUS command is required by SHE.

CMD_CANCEL

As already stated, execution of the CMD_CANCEL command is not started using the regular SHE command register but instead with a dedicated command cancel register (SHE_CMDCANCEL). To cancel an ongoing command the user has to set the SHE_CMDCANCEL:CANCELREQ bit to '1'. Once this is detected by the SHE control logic, the internal processing of the CMD_CANCEL command starts. This is indicated by the change of error code in the error code register SHE_ERC for the sequential command (SHE_ERC:CMDMAIN) and command cancel (SHE_ERC:CANCELMAIN) to ERC_INVALID. Furthermore, similar to the processing of a sequential command, the busy flag (SHE_STATUS:BUSY) is kept high and the done flag (SHE_STATUS:DONE) is kept low during the processing of the CMD_CANCEL command. Additionally, the command register SHE_CMD is locked for writing while CMD_CANCEL is being processed, i.e. while SHE_STATUS:BUSY is high.

Once the internal processing is finished, the SHE control logic updates SHE_ERC and sets the SHE_STATUS:DONE flag. The error code for the canceled sequential command (SHE_ERC:CMDMAIN) is set to ERC_CANCELED and the error code for the command cancel (SHE_ERC:CANCELMAIN) is set to a valid value defined by the SHE Functional Specification. The user application has to read the error code register using 32-bit data access in order to complete command execution. The SHE_STATUS:BUSY and SHE_STATUS:DONE flags are then cleared. Figure 43-34 illustrates the timing flow for the execution of the CMD_CANCEL command.

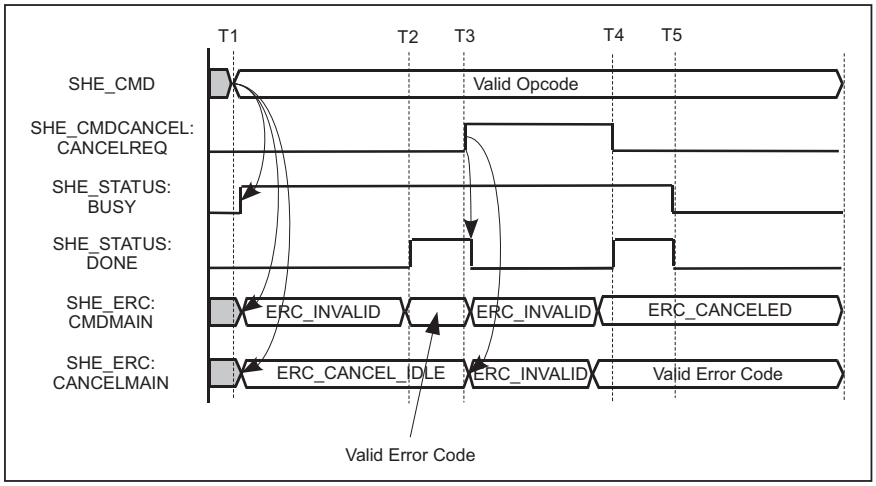
Figure 43-34. Execution of CMD_CANCEL command



The cancel request for the command which was started at moment T1 is detected at moment T2. Internal processing of CMD_CANCEL is finished at moment T3 and the error code register is read at moment T4. A new command can be started at any time after T4, for example at moment T5.

Figure 43-35 illustrates the timing flow for the execution of a CMD_CANCEL command issued after the end of the internal data processing in SHE, i.e. when the valid error code (SHE_ERC:CMDMAIN) is available and the done flag (SHE_STATUS:DONE) is high.

Figure 43-35. Execution of CMD_CANCEL command issued after the previous command has completed

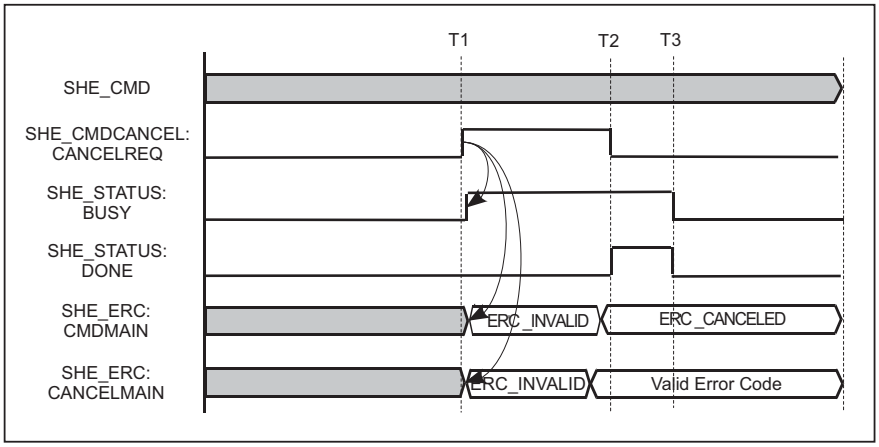


The internal processing of the sequential command, which was started at moment T1, is completed at moment T2. The cancel request is issued at moment T3. Both error codes (SHE_ERC:CMDMAIN and SHE_ERC:CANCELMAIN) are reset and the done flag (SHE_STATUS:DONE) is then cleared by the hardware. Internal processing of CMD_CANCEL is completed at moment T4 and the error code register is read at moment T5.

The CMD_CANCEL command can also be issued when no command is being executed. The appropriate timing diagram is shown in Figure 43-36. The cancel request is issued at moment T1. Both error codes, SHE_ERC:CMDMAIN and SHE_ERC:CANCELMAIN, are reset, the SHE_STATUS:BUSY flag is set and the SHE_STATUS:DONE flag is kept low by the hardware. Internal processing of CMD_CANCEL is completed at moment T2.

Note: Only the error code for CMD_CANCEL (SHE_ERC:CANCELMAIN) is updated. The error code register for the sequential command (SHE_ERC:CMDMAIN) keeps its value (ERC_INVALID).

Figure 43-36. Execution of CMD_CANCEL command when no command execution is ongoing.



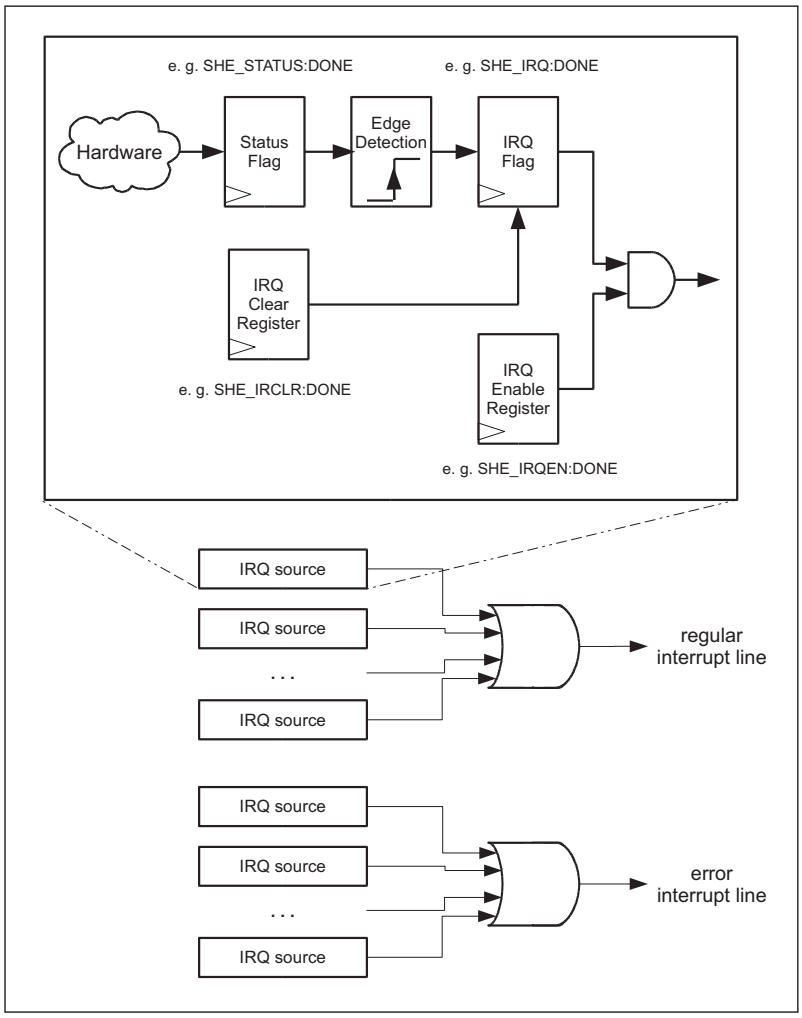
Interrupt request generation

In order to support interrupt-based application design, SHE provides two separate interrupt request lines to signal changes to its internal state. The first interrupt line is used for regular interrupts and is triggered by regular events that are expected during command processing. For example, the rising of the done status flag (SHE_STATUS:DONE) triggers the regular interrupt request. The second interrupt line is used to signal extraordinary events, i.e. errors.

Both interrupts are shared interrupts, i.e. each interrupt line has several interrupt sources. To distinguish between interrupt sources the user has to perform a read access to the SHE Interrupt Request register SHE_IRQ. This register represents all interrupts available in SHE and is divided into two parts: regular and error interrupts. All interrupts can be enabled and cleared independently of each other using the SHE_IRQEN and SHE_IRQCLR registers respectively. The architecture of the interrupt circuitry used in SHE is shown in Figure 43-37 In general it consists of the following parts:

- Status flags that represent the actual state and which change accordingly with it. For example, the DONE flag in the SHE_STATUS register.
- Edge detection logic. The rising edge of the status flag sets the interrupt request flag.
- Interrupt request flags representing pending interrupt request. Each flag is set by the rising edge of the corresponding status flag and remains high until it is cleared by the software, e.g. the DONE flag in SHE_IRQ register.
- IRQ enable bits. If written to ’1’, the corresponding interrupt request stored in the SHE_IRQ register is propagated to the appropriate interrupt line. For example, the SHE_IRQEN:DONE bit enables the generation of a SHE interrupt for the SHE_STATUS:DONE bit.
- The interrupt request flag should be cleared by the software during the execution of its interrupt service routine. This is achieved by writing ’1’ to the appropriate bit in the interrupt request clear register, i.e. SHE_IRQCLR:DONE.

Figure 43-37. Block diagram of SHE interrupt generation circuitry



Note: All IRQ source circuits are identical to the one shown in detail.

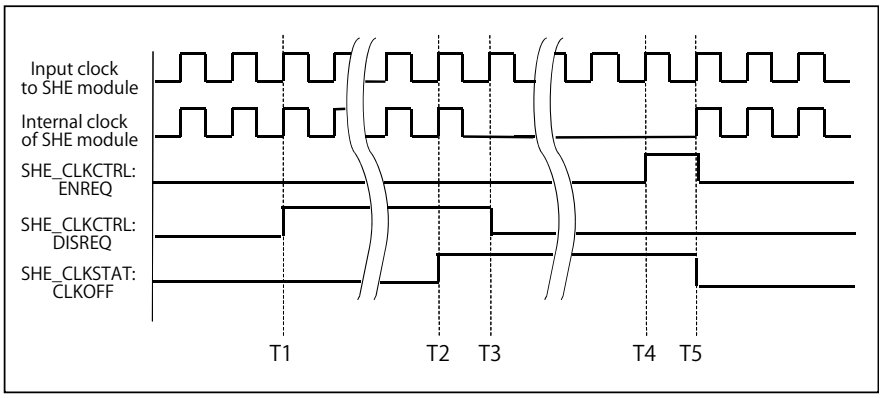
As already mentioned, several interrupt sources are combined and the resulting shared interrupt lines (one for regular and one for error interrupts) are then connected to the interrupt controller.

43.3.1.1 Power saving functionality

Clock gating for SHE

The power consumption of SHE can be reduced by disabling its internal clock. Enabling/disabling the SHE clock can be performed using the SHE_CLKCTRL register. The current status of the clock is given by the SHE_CLKSTAT register. The procedure for clock gating is presented in Figure 43-38.

Figure 43-38. SHE clock disabling/enabling



In order to disable the SHE clock the user has to set the SHE_CLKCTRL:DISREQ bit to '1' (T1). Please note that access to the SHE_CLKCTRL register can occur at an arbitrary point in time, but the SHE clock can only be disabled if SHE is in an idle state (i.e. SHE_STATUS:BUSY is '0'). Hence the pending clock disable request is kept high until SHE moves to the idle state and performs the internal operations required for clock gating. The CLKOFF bit in the SHE_CLKSTAT register is then set (T2). The pending disable request SHE_CLKCTRL:DISREQ is automatically cleared by the SHE hardware, and the SHE clock is disabled on the next cycle of the input clock (T3).

The clock is enabled by writing '1' to the SHE_CLKCTRL:ENREQ bit (T4). SHE_CLKCTRL:ENREQ and SHE_CLKSTAT:CLKOFF are cleared after which the internal clock is enabled (T5).

External debugger detection is active even if the SHE clock is switched off. More precisely, the external debugger will be detected if it is attached during SHE power-saving mode, and the SHE_STATUS:EXTDEBUGGER bit will be set after the SHE clock has been enabled.

Note: Only the SHE_CLKCTRL and SHE_CLKSTAT registers are accessible if the internal SHE clock is disabled. Accessing other SHE registers will generate a bus error response.

System wide clock/power gating

The status register bit SHE_STATUS:BOOTOK is automatically cleared whenever the CPU or a memory of the MCU enters power saving mode. This means for a FCR4 series device, SHE is notified if the clock is gated for the following modules (the appropriate clock name is given in parentheses):

- Cortex-R4 CPU (CLK_SYS_PD3)
- TCFLASH (CLK_SYS_PD3 or CLK_HPM_PD3)
- EEFLASH (CLK_SYS_PD3)
- System RAM (CLK_HPM_PD2)
- Retention RAM (CLK_CFG_PD4)
- Ethernet RAM (CLK_PERI4_PD2)
- Graphics clock of the subsystem, video RAM (CLK_GFX_PD5)

Furthermore SHE is notified if power domains PD4 or PD5 are powered down.

Note: Only falling edges of the clock/power enable signals for the modules listed above are taken into account. Thus the SHE_STATUS:BOOTOK bit will be cleared only when the clock/power is disabled.

43.3.1.2 Command interface access issues

Access to the SHE command Interface is protected through a PPU. In other words reading and writing in user mode are only possible if the PPU is appropriately configured. Privileged accesses are always allowed.

A master ID filter is also implemented. This means that only two specified masters, the CPU and debugger, are allowed to access the SHE command interface.

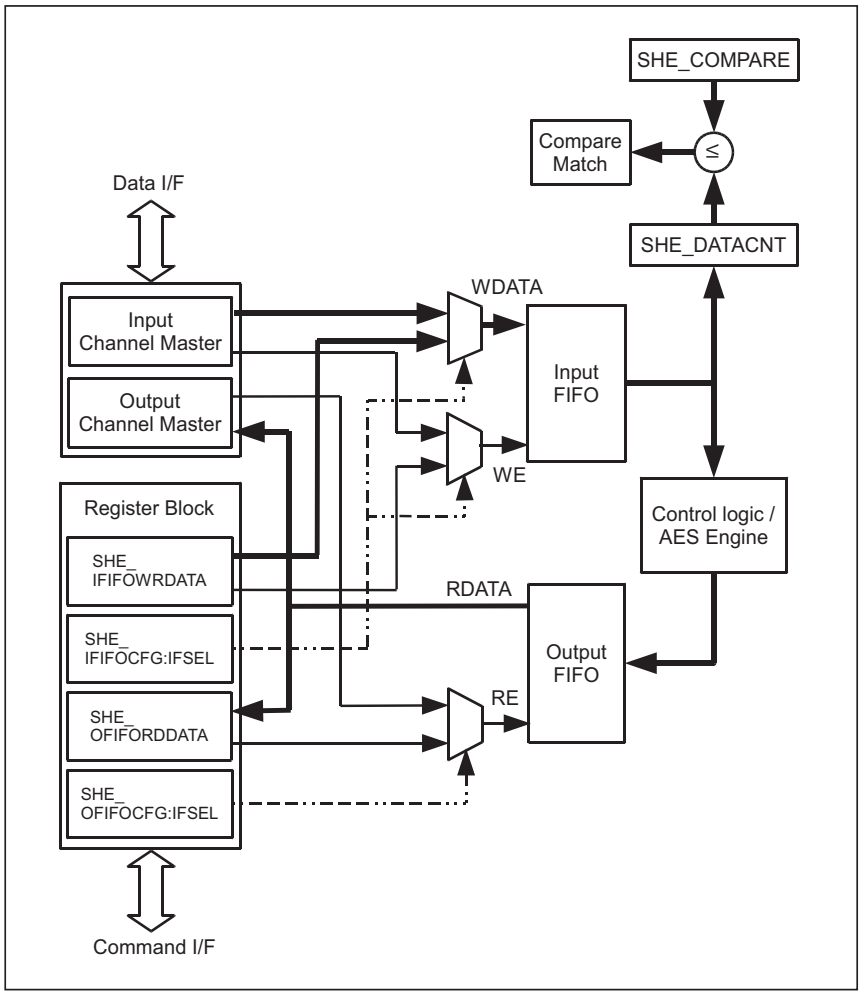
Since the command interface can also be utilized for the transfer of parameters and data used for processing, special registers exist for accessing the write port of the input FIFO (refer to the Input FIFO Write Data Register (SHE_IFIFOWRDATA0~31) description) and read port of the output FIFO (refer to the Output FIFO Read Data Register (SHE_OFIFORDDATA0~31) description). An error interrupt flag is set (refer to the Interrupt Request Register (SHE_IRQ) and bits IFIFOWRERR and OFIFORDERR) if write access to a full input FIFO or read access to an empty output FIFO is performed over the command interface. Moreover, an error response on the AHB bus is generated in the event of a write access to SHE_IFIFOWRDATA if the input FIFO is locked (see Section n "Input FIFO"). Hence the state of the FIFOs has to be checked before performing the accesses.

43.3.2 Operation of the data interface

All data required for the execution of a single SHE API command as well as results of commands can be transferred over the SHE data interface. This interface is controlled by two data master modules which transfer data to/from the attached FIFOs. The data flow diagram is shown in Figure 43-39.

By default the data transfer is done over the command interface, but alternatively the data can also be transferred over the data interface using data masters. Refer to SHE_IFIFOCFG:IFSEL, SHE_OFIFOCFG:IFSEL, SHE_IFIFOWRDATA and SHE_OFIFORDDATA register descriptions.

Figure 43-39. SHE data flow diagram



Feature List

The features of the data interface are:

- A 64-bit AXI master interface
- Input and output channels for independent read and write data transfers
- Only incremental bursts to 64-bit aligned addresses are supported
- Direct memory access capability, which covers the complete address range of the MCU (32-bit)
- Configuration settings lock during transfer
- FIFOs for synchronisation and data transfer between several parties with different data-path widths
- Support for interrupt-based software design

Operation of data master modules

The input and output channel masters are responsible for performing data transfers to and from SHE. They can perform independent read and write transfers between different memory locations of the MCU and SHE. In order to increase data transfer performance, both masters always try to perform burst transfers of the maximal possible length.

Both data masters have the same set of configuration registers:

- Start address in memory where the master has to read data from or store to. Refer to the Input Channel Master Start Address Register (SHE_IMSTADDR) and Output Channel Master Start Address Register (SHE_OMSTADDR) descriptions. The address has to be 64-bit aligned. The start address should also be aligned to the 128-byte boundary if the transfer is supposed to cross the 4KB address boundary. Otherwise at least one short burst transfer (on reaching the 4KB boundary) will be performed.
- The total number of transfers required for a single command. Refer to the Input Channel Master Data Transfer Counter (SHE_IMSTCNT) and the Output Channel Master Data Transfer Counter (SHE_OMSTCNT) register descriptions. This value, which has to be preloaded by software, is the number of 64-bit double words that have to be transferred. This value is decremented with every transfer and represents the amount of remaining data transfers.
- Data transfer start trigger. Refer to Input Channel Master Start Trigger (SHE_IMSTSTART) and Output Channel Master Start Trigger (SHE_OMSTSTART) registers. The transfer can only be started if the master idle flag (IMSTIDLE or OMSTIDLE in the SHE_MSTSTATUS register) is high. Trying to start the transfer while the master idle flag is low has no effect. Once started the transfer proceeds as long as the number of remaining transfers (SHE_*MSTCNT) is greater than zero.

The configuration settings can be written before or after the start of the command (write access to the SHE_CMD register). The data transfer itself (through the SHE_*MSTSTART register) should only be started after the start of the command. Otherwise the data which may have already been transferred into the input FIFO will be discarded because the FIFO is flushed opon the start of the command.

Note:

If the start of a new data transfer is requested (i.e. a write access to the SHE_*MSTSTART register) and the interface setting for the appropriate FIFO is ‘Command Interface’ - refer to either the Input FIFO Configuration Register (SHE_IFIFOCFG) or Output FIFO Configuration Register (SHE_OFIFOCFG) - which is in fact an error condition, the value of the SHE_*MSTSTART register remains zero but the SHE_IRQ:*MSTIFSELERR error interrupt flag is set instead and the transfer is not started.

To prevent the manipulation of settings, the appropriate configuration registers are automatically locked after starting a data transfer. An enabled lock is signaled through the IMSTLOCK or OMSTLOCK bit in the SHE_MSTSTATUS register. Writing access to locked registers produces a bus error response. The lock is released either if the required amount of data has been transferred (SHE_*MSTCNT is zero) or if the data transfer was canceled in case of an error or CMD_CANCEL. The complete overview of lockable registers is given in Table 43-34.

Table 43-34. Lockable Registers List

	Input Channel Master	Output Channel Master
Lock bit	SHE_MSTSTATUS:IMST- LOCK	SHE_MSTSTATUS:OMSTLOCK
Locked Registers	SHE_IFIFOCFG:IFSEL SHE_IMSTADDR SHE_IMSTCNT	SHE_OFIFOCFG:IFSEL SHE_OMSTADDR SHE_OMSTCNT

As already mentioned in Section n "Sequential command execution", the decision concerning which interface should be selected for data transfer depends on the amount of data. The software may use only one, both or even no data masters. Also a combination of command interface and data interface is possible; the parameters can be transferred through the command interface and the subsequent data required for processing can be transferred by the input channel master. The results can also be read through the command interface or by the output channel master.

Behavior in the event of a CMD_CANCEL command

Both data masters feature special stop request input signals, which are controlled by the SHE control logic and can be set in order to cancel an ongoing data transfer. These signals will be asserted in the event of the pending CMD_CANCEL command.

If the stop request is detected by a data master, it completes any ongoing burst but cancels all subsequent ones. The ongoing burst is completed as follows: the input channel master reads the remaining data from the AXI bus but discards it and doesn't write it into input FIFO. The output channel master stops reading the output FIFO and writes the default data to the bus (0x0...0), but AXI byte strobes are disabled.

Note: The transfer counter (SHE_*MSTCNT) only counts regular transfers, i.e. the counter value is not changed if the data transfer has been canceled and "dummy" transfers are performed. After finishing the current burst, the master moves to an idle state and asserts the appropriate master idle flag (SHE_MSTSTATUS:*MSTIDLE). No new data bursts will be started until a new command is issued. The transfer counter (SHE_*MSTCNT) value is kept, allowing the software to determine the amount of data transferred.

Error Handling

Two kinds of errors can occur during the operation of the data masters:

1. Write access from the input channel master to the locked input FIFO.

   The input FIFO is locked when the required amount of data has been transferred (see 43.3.3 Operation of the FIFO unit). However, misconfiguration may cause the input channel master to execute write accesses to the input FIFO. In this case an error interrupt flag (SHE_IRQ:IFIFOLOCKERR) is set to notify the software of this error.

   Upon error detection, the input data master behaves in the same way as if it were processing a stop request (see 'Behavior in the event of a CMD_CANCEL command'), i.e. an ongoing burst is completed without transferring the data to the input FIFO and all subsequent bursts are canceled. The output data master is not affected in this case and continues its operation.

2. Bus error response.

   An AXI slave can generate an error response in the event of an error during data transfer. Since SHE cannot resolve bus errors, the software should be informed about it. This will be done by setting the status flags (SHE_MSTSTATUS:IMSTERR, SHE_MSTSTATUS:OMS- TERR) and appropriate error interrupt flags - SHE_IRQ:IMSTERR for the input channel master and SHE_IRQ:OMSTERR for the output channel master. Besides the interrupt, the AXI response code (IMSTERRRESP and OMSTERRRESP in SHE_MSTSTATUS) and the error address (SHE_IMSTERRADDR and SHE_OMSTERRADDR) are stored.

According to the AXI protocol definition for a read transaction, the slave can give different responses for different transfers within a burst. Hence, on receiving an error response the input channel master behaves in the same way as if it were processing a stop request (see 'Behavior in the event of a CMD_CANCEL command'), i.e. an ongoing burst is completed without transferring the data to the input FIFO, and all subsequent bursts are canceled.

According to the AXI protocol definition for a write transaction, there is just one response given for the entire burst and not for each data transfer within the burst. Hence, on receiving an error response the output channel master does not have to complete any ongoing burst and just cancels any outstanding transfers.

Error responses on both data channels are also signaled to the SHE control logic, which will cancel the current command execution and return an error code thereupon.

System integration issues

- Memory protection

  For safety reasons both masters have a Memory Protection Unit (MPU) in order to protect the memory areas of the MCU. For details on MPU configuration, refer to 6. Memory Protection Unit for AXI of the FCR4 Cluster Series Hardware Manual.

- Instruction flash addressing

  The instruction flash (TCFLASH) is divided into three parts (beginning at address zero): large sectors, reserved area and small sectors. For access to the reserved area within the TCFLASH an error response is generated, which would cause the SHE input master to abort a transfer and trigger an error interrupt. The consequence for the application is that a vali-

dation of the complete TCFLASH would be impossible. To overcome this, the input channel master jumps from the last address in the large sectors part to the first address of the small sectors part.

Similar behavior is shown if the input channel master reaches the end of the small sectors part. In this case an address wrap around occurs and read accesses continue at address zero. This feature together with the previous one allows complete TCFLASH addressing independently of the starting location (128-bit aligned).

Note:

The approach described above only applies if a valid address in the TCFLASH (not a reserved area) is given in the SHE_IMSTADDR register. This means if SHE_IMSTADDR points to a reserved area of the TCFLASH, no special action is performed and the usual Error Handling is done.

Notes on the operation of the data masters

The data interface implements the AXI protocol using a 32-bit address and 64-bit data buses. Input and output data transfers can be carried out simultaneously by performing up to 16-beat long incremental bursts. This means up to 16 x 64 = 1,024 bits of data can be transferred in one burst. In order to simplify the design, no unaligned accesses are allowed, i.e. all addresses have to be aligned to the 64-bit boundary.

The number of transfers in a single burst is determined by the value of the total transfer counter (see 43.2.4.3 Input Channel Master Data Transfer Counter (SHE_IMSTCNT) - SHE_IMSTCNT and 43.2.4.4 Output Channel Master Data Transfer Counter (SHE_OMSTCNT) - SHE_OMSTCNT). While this value is greater than or equal to 16, the master issues bursts with a maximum length of 16. Once the number of remaining transfers becomes less than 16, this value will be taken to perform the last burst.

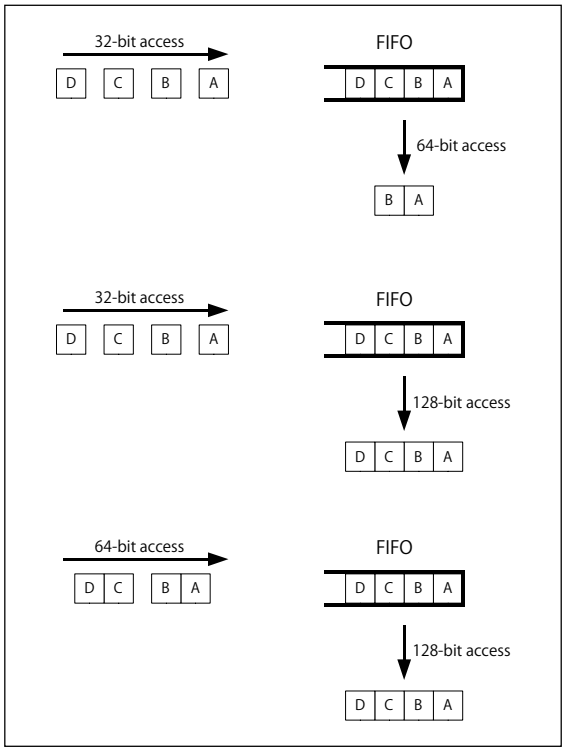
Due to the limitation of the AXI protocol, which cannot perform bursts over the 4KB address boundary, the master has to reduce the size of the burst if it is about to cross the 4KB boundary. All consecutive bursts will be performed using the approach described in the previous paragraph. FIFO load information is also taken into account in order to maximize the efficiency of the data transfer. Incoming data transfers are only started if enough free space is available in the input FIFO. Outgoing data transfers are only started if the output FIFO contains enough data for a complete burst.

The data master performs no manipulation on the transferred data; it is ensured that the data which is delivered to the internal AES engine has the same content and order as it was provided by the user in the source memory location.

43.3.3 Operation of the FIFO unit

All data transfers to and from the SHE module are done over two FIFO memories. Additionally they are used for synchronisation between several parties with different data-path widths. For example, the data can be written to the input FIFO via the command interface using 32-bit accesses or via the data interface using 64-bit accesses. The same is also valid for the output FIFO, which can be read either by a 32-bit or 64-bit access. Hence, to ensure the correct synchronisation and data order, the least significant part of the data has to be transferred first. More precisely, if the transfer of a data item has to be split into several transfers, the order shown in Figure 43-40 must be kept.

Figure 43-40. FIFO data ordering



With reference to Figure 43-40:

1. If a 64-bit double word has to be written/read using two 32-bit accesses, the least significant word, i.e. bits[31:0], has to be transferred first followed by the most significant word, i.e. bits [63:32].
2. If a 128-bit quad word has to be written/read using four 32-bit accesses, the least significant word, i.e. bits[31:0], has to be transferred first followed by bits[63:32] and bits[95:64]. The most significant word, i.e. bits[127:96], is transferred last.
3. If a 128-bit quad word has to be written/read using two 64-bit accesses, the least significant double word, i.e. bits[63:0], has to be transferred first followed by the most significant double word, i.e. bits[127:64].

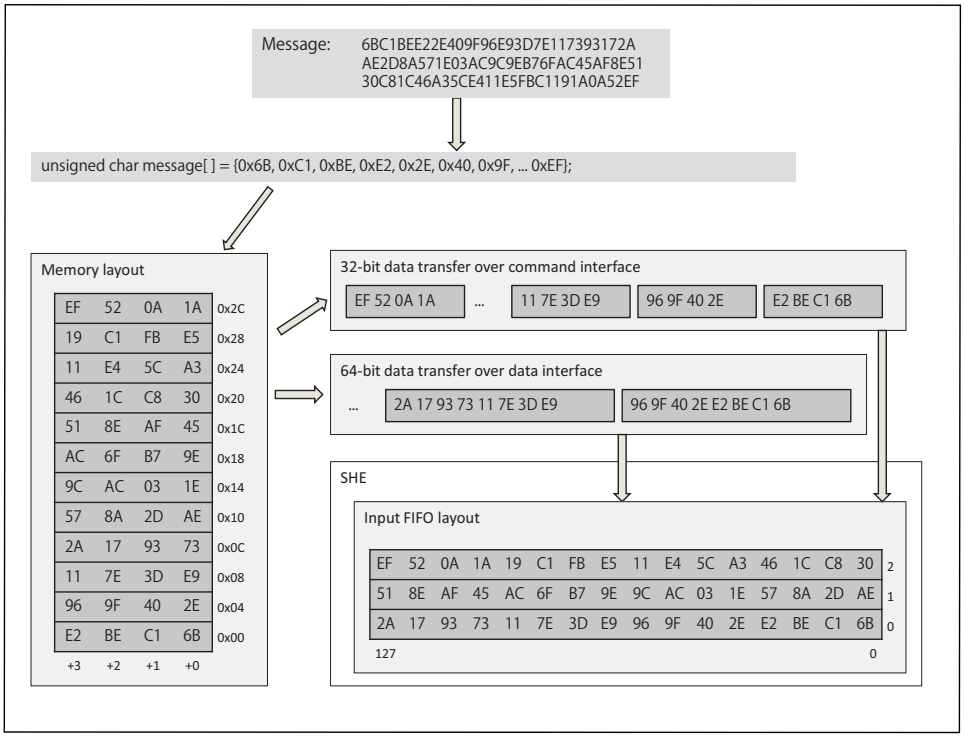
Data ordering for the command parameters

Command parameters given in Table 43-33 must be transfered according to one of the following ways depending on the parameter width:

1. 64-bit wide parameters must be represented in the native little endian byte order of FCR4 host system and, as described in the 43.3.1 Operation of the command interface, the least significant part of the parameter has to be transfered first. Forexample if the KEY_ID parameter is 4 (0x0000000000000004) and it will be transferred to SHE using two 32-bit accesses, the data for the first access will be 0x00000004 and for the second will be 0x00000000.
2. Parameters wider than 64-bits must be represented in the big endian ordering of bytes and the most significant byte (MSB) must be transfered first. Please refer to the Figure 43-41 for an example showing data transfer of a 384-bit large message. Since cryptographic messages are usually stored using byte arrays, the MSB of the message will be stored in the memory at the offset zero resulting in the memory layout given in the example. For the processing in SHE the mes-

sage can be transfered either through the command interface (usind SHE_IFIFOWRDATA registers) or through the data interface (using input data master). The resulting layout of the input FIFO of SHE represents the message with byte-wise swapped 128-bits data blocks. This swapping is then compensated by the SHE hardware during the internal processing. Note that the results of the SHE processing which are wider than 64 bits are also represented in the big endian ordering of bytes and are available in the output FIFO of SHE in the byte swapped way.

Figure 43-41. FIFO data ordering for parameters wider than 64 bits



Input FIFO

The input FIFO of SHE is used for the transfer of parameters and data required for SHE command execution. It can be written automatically through the input data master or manually by the user through the command interface - refer to the Input FIFO Write Data Register (SHE_IFIFOWRDATA0~31) description. The input FIFO has a capacity of 1,536 bits, which is sufficient to store one and a half bursts from the input data master. The size of free space in the input FIFO can be determined by reading the SHE_FIFOLOAD:IFIFOFREE status register.

The input FIFO provides a programmable write threshold setting - refer to the Input FIFO Configuration Register (SHE_IFIFOCFG) description. If the amount of stored data words is less than the stored value, a status flag (SHE_FIFOSTATUS:IFIFOWRTH) and an interrupt flag (SHE_IRQ:IFIFOWRTH) are set.

SHE has to be able to recognize that the amount of data required for the execution of a SHE API function has been transferred through the input FIFO. For this purpose two specific registers are introduced: SHE_COMPARE and SHE_DATACNT. The former contains the required amount of data and the latter contains the amount of data actually transferred. According to the SHE specification, the largest possible data stream size for a function is 2^{64} bits. And since both registers represent data amounts in terms of 32-bit words, their size is specified to 59 bits.

SHE_DATACNT is an up-counter and is automatically reset to zero every time the opcode for a new SHE API function is written to the SHE_CMD register. Every read access to the input FIFO increases the value of the counter. When the counter reaches the value of the SHE_COMPARE register, the status bit SHE_FIFOSTATUS:COMPAREMATCH and an interrupt request bit SHE_IRQ:COMPAREMATCH are immediately set. This transition of the COMPAREMATCH bit is referenced to as a compare match event.

SHE_COMPARE is reset by the hardware to the maximal value (2^{59} - 1) every time the opcode for a new SHE command is written to the SHE_CMD register. This approach avoids an immediate compare match event in case of low reset values

(including zero value). In most cases the SHE_COMPARE register is configured through the SHE control logic according to selected command and/or parameter values and cannot be changed during command execution. The only exception to this rule is two SHE commands CMD_ENC_CBC and CMD_DEC_CBC. In this case the amount of processing data is not always known (i.e. a priori) and command execution is started using the default value of SHE_COMPARE. Once the required amount of data is known, the software updates the SHE_COMPARE register.

According to the functional specification of SHE, some commands require notification in case the amount of delivered data exceeds that required. This notification will be stored in the error code register and can be read by the software. But since the error code has to be read by software before a new command can be written, there is a time window from reading the error code to writing the new command where new data could enter the input FIFO without being recognized by the software (error code already read). In order to avoid this situation, the write port of the input FIFO has to be locked at the time when all input data have been transferred until the next command is written. This means the lock is enabled immediately after the compare match event and disabled upon writing new opcode to the SHE_CMD register. If any write access to the locked input FIFO is performed by the input data master or through the command interface, an error interrupt flag (SHE_IRQ:IFIFOLOCKERR) is set.

Output FIFO

The output FIFO of SHE is used to transfer the results of SHE commands. It can be read automatically by the output data master or manually by the user through the command interface - refer to the Output FIFO Read Data Register (SHE_OFIFORDDATA0~31) description. The capacity of the output FIFO is equal to 1,536 bits, which is sufficient to store one and a half bursts for the output data master. The load status of the output FIFO can be determined by reading the SHE_FIFOSTATUS:OFIFOLOAD status bit.

The output FIFO provides the programmable read threshold setting (see SHE_OFIFOCFG:RDTHRESHOLD bit description). If the amount of stored data words is greater than the stored value, a status flag (SHE_FIFOLOAD:OFIFORDTH) and an interrupt flag (SHE_IRQ:OFIFORDTH) are set.

To avoid an inconsistent state in case of excess data contained in FIFOs, they are both cleared automatically at the beginning of the command. More precisely, the FIFOs are cleared by a writing access to the command register SHE_CMD while SHE is not busy.

43.4 Notes on using SHE

This section lists some issues which has to be taken care of when using the SHE module. Programmers should read these guidelines before programming the SHE module.

43.4.1 General notes on using SHE

SHE is connected to the same reset and the same clock signals as the CPU and the EFLASH. The only difference is that the clock of SHE cannot be switched off in the system controller and is always enabled. However SHE allows local clock gating (refer to 41.3.1.1 Power saving functionality).

Note. Since secure key storage is implemented in EEFLASH, the EEFLASH clock must be enabled when SHE functionality is required.

43.4.1.1 Notes on using SHE commands

CMD_SECURE_BOOT

Secure boot process is automatically started after every reset. Secure boot settings i.e. secure boot mode, start address and size if the protected memory are stored in the BDR area of the TCFLASH. The Boot ROM code reads these settings and afterwards evaluates them and configures the input data master of SHE and starts execution of the CMD_SECURE_BOOT.

Note that the user application is started only after the execution of CMD_SECURE_BOOT inside SHE has begun. Hence from the point of view of user application some SHE registers do not have their reset values as defined in this document. Depending on secure boot settings following registers can be effected:

- SHE_CMD
- SHE_STATUS
- SHE_ERC
- SHE_IMSTADDR
- SHE_IMSTCNT
- SHE_IMSTSTART
- SHE_IFIFOCFG
- SHE_COMPARE0~1
- SHE_MSTSTATUS
- SHE_FIFOSTATUS
- SHE_FIFOLOAD
- SHE_DATACNT0~1

Depending on secure boot mode, the secure boot process may not have finished when entering the application. The application software or SHE driver shall consider that SHE might still be busy and should either wait for its completion or cancel the process.

Autonomous bootstrap configuration of the secure boot process

If the secure boting has been personalized during the same boot process, the following is valid for the BOOT_MAC memory slot:

- None of the flags are set
- The counter is equal to 1

Memory used for secure boot

Though all of the memories of MCU can be used for the secure boot process, it is recommended to use only the instruction flash (TCFLASH), since write access to it is monitored by SHE. This means a write access to instruction flash during the secure boot leads to the secure boot failure.

Secure boot settings

Secure boot start address should be alligned to 64-bit boundary (lower 3 bits are ‘0’). Otherwise it will be automatically alligned by the Boot ROM code during the evaluation of the secure boot settings. Note that only the start address which is used for setting up the input data master of the SHE is modified, not the address in BDR which is the entry address to the application. The amount of data which is transfered to SHE for the CMD_SECURE_BOOT is always a multiple of 128-bits. If the secure bit size parameter stored in BDR is not a multiple of 128-bits, following applies ‘data size of input data master configuration’ =ciel (size from BDR’ / 16) * 2. The ammount of valid data is still given by the secure boot size parameter in the BDR.

If secure boot start address and/or secure boot size are modified by the Boot ROM, the secure boot process may try to access non-initiallized memory regions before or after the authenticated area. If such access causes bus error, secure boot process will fail.

If secure boot start address and/or secure boot size are modified by Boot ROM during successful personalization (self-learning of BOOT_MAC) and the secure boot settings in the BDR are not changed by the user anymore, same adjustment applies to secure boot process during next reset cycles and succesful secure boot is possible.

Furthermore this automated adjustment of the secure boot settings must be kept in mind if the BOOT_MAC is calculated offline and loaded into SHE using CMD_LOAD_KEY. Other wise the BOOT_MAC calculated by SHE may not be equal to the reference BOOT_MAC and the secure boot will fail.

CMD_LOAD_KEY

The initial (empty) value of the non-volatile slots is: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF. Special care has to be taken if the memory update command fails, i.e the error code register SHE_ERC:CMDMAIN contains an error code other than ERC_NO_ERROR. In such a case the user should check whether the old or the new version of the key is valid before repeating the update. If an error occurs before or during the physical write operation to the memory, the old value is still valid. If an error occurs after the physical write operation to the memory, the new value is valid.

SHE stores keys in a redundant way which is intended to cover aging and transient errors. Occurance of such errors in the memory slot during the write by CMD_LOAD_KEY cause the command to fail despite redundancy and the NVM state is not changed. However in some rare cases such errors are accepted by CMD_LOAD_KEY and th memory slot is updated. In that case the command CMD_LOAD key will succeed (SHE_ERC:CMDMAIN contains ERC_NO_ERROR), but the user will be notified by the warning code (other than ERC_INVALID) in the extended error code register (SHE_ERC:CMDEXTD) and can react by repeating the memory update procedure.

43.4.1.2 SHE Used Error Codes/Used Cancel Codes

Table 43-35. SHE Used Error Codes

Cancel Error	Command Error	Main Error	Description
0x0010	0x0001	ERC_NO_ERROR	Command was normally completed.
See Table 43-36	0x000E	ERC_CANCELED	The current command was aborted by SHE_CMD_CANCEL:CANCELREQ.
0x0010	0x010D	ERC_GENERAL_ERROR	AES engine could not be stopped after a preceeding error.
0x0010	0x020D	ERC_GENERAL_ERROR	Internal register access invoked a bus error.
0x0010	0x4D06	ERC_NO_SECURE_BOOT	TCFLASH was written during Secure Boot.
0x0010	0x4506	ERC_NO_SECURE_BOOT	Secure boot is disabled by the empty secure boot key.
0x0010	0x2f03	ERC_KEY_NOT_AVAILABLE	While a boot protected key was used, the BOOTOK flag was cleared, e.g. by clock switching.

Table 43-35. SHE Used Error Codes

Cancel Error	Command Error	Main Error	Description
0x0010	0x3003	ERC_KEY_NOT_AVAILABLE	While a debugger protected key was used, the EXTDEBUGGER flag was set due to the connecting debugger.
0x0010	0x390D	ERC_GENERAL_ERROR	The requested command is not available in the current state or mode.
0x0010	0x370D	ERC_GENERAL_ERROR	The AXI channel master stopped host-to- SHE transfer due to a bus error.
0x0010	0x360D	ERC_GENERAL_ERROR	The AXI channel master stopped SHE-to- host transfer due to a bus error.
0x0010	0x1904	ERC_KEY_INVALID	Invalid SHE slot number command argument.
0x0010	0x1004	ERC_KEY_INVALID	The RAM key to be exported was loaded as a plain key.
0x0010	0x1E09	ERC_RNG_SEED	The command requires initialized RNG but CMD_INIT_RNG was not executed before or was discarded by CMD_DEBUG.
0x0010	0x3A0D	ERC_GENERAL_ERROR	Illegal device test mode request in SHE operation mode.
0x0010	0x1C04	ERC_KEY_INVALID	Key update protocol violated by an invalid key ID or an authentication ID.
0x0010	0x2708	ERC_KEY_UPDATE_ERROR	Key update violated by unmatched UID.
0x0010	0x1B08	ERC_KEY_UPDATE_ERROR	Key update violated by unmatched M3.
0x0010	0x1308	ERC_KEY_UPDATE_ERROR	Key update protocol violated by a wildcard UID despite the wildcard being disabled for the slot.
0x0010	0x4E0D	ERC_GENERAL_ERROR	CMD_INIT_RNG failed due to missing TRNG calibration data.
0x0010	0x030D	ERC_GENERAL_ERROR	Command forbidden due to invalid device state or the NVM state.
0x0010	0x040D	ERC_GENERAL_ERROR	Unknown command request code.
0x0010	0x190A	ERC_NO_DEBUGGING	CMD_DEBUG failed due to invalid authentication.
0x0010	0x1107	ERC_KEY_WRITE_PROTECTED	CMD_DEBUG failed due to at least one write protected or invalid SHE slot.
0x0010	0x170D	ERC_GENERAL_ERROR	At any command end there are still unprocessed data in the input FIFO if an extra argument was falsely provided to SHE.
0x0010	0x3C0D	ERC_GENERAL_ERROR	This is a special asynchronous error code which may occur independently of the normal command end if the ECC double bit errror in the SHE RAM does not allow further SHE operations. If it occurs, the lower byte of SHE_STATUS is not updated but SHE_STATUS:FATAL is set. SHE is halted.
0x0010	0x400D	ERC_GENERAL_ERROR	MCU device mode was changed on-the-fly during the current SHE operation mode. SHE is halted.
0x0010	0x420D	ERC_GENERAL_ERROR	Illegal combination of device mode signals and SHE mode signals. SHE is halted.

Table 43-35. SHE Used Error Codes

Cancel Error	Command Error	Main Error	Description
0x0010	0x3D0D	ERC_GENERAL_ERROR	Invalid NVM state found during SHE start- up. SHE is halted.
0x0010	0x3B0D	ERC_GENERAL_ERROR	Despite SHE being disabled due to the NVM state and device mode, the device mode changes again. SHE remains stopped.
0x0010	0x410D	ERC_GENERAL_ERROR	The device mode changes again despite SHE being in test mode. SHE remains stopped.
0x0010	0x430D	ERC_GENERAL_ERROR	NVM is blank. Fabrication is not complete. SHE is halted.
0x0010	0x3E0D	ERC_GENERAL_ERROR	The device mode changes despite SHE being in operation. SHE is halted.
0x0010	0x340C	ERC_MEMORY_FAILURE	Internal NVM address is out of range. NVM operation skipped.
0x0010	0x310C	ERC_MEMORY_FAILURE	Reading NVM showed there was insufficient redundancy to rely in data
0x0010	0x4A0D	ERC_GENERAL_ERROR	Slot ID is out of range. Internal error which should never occur.
0x0010	0x350D	ERC_GENERAL_ERROR	An active NVM slot revision was not completely loaded. This error should never occur.
0x0010	0x330C	ERC_MEMORY_FAILURE	Failed read verfication of a word just written to the NVM.
0x0010	0x0B03	ERC_KEY_NOT_AVAILABLE	A boot protected key is used even though secure boot had failed (BOOTOK == 0).
0x0010	0x1A08	ERC_KEY_UPDATE_ERROR	Key update protocol violated by invalid key counter.
0x0010	0x0F03	ERC_KEY_NOT_AVAILABLE	A debugger protected key is used despite external debugger being connected.
0x0010	0x1204	ERC_KEY_INVALID	The key usage for encryption/decryption or MAC calculation/verification does not match the slot flag.
0x0010	0x1806	ERC_NO_SECURE_BOOT	Invalid sequence of secure boot commands CMD_SECURE_BOOT, CMD_BOOT_OK and CMD_BOOT_FAILURE.
0x0010	0x1F0C	ERC_MEMORY_FAILURE	A bus error occured when accessing the NVM private bus.
0x0010	0x210D	ERC_GENERAL_ERROR	A double bit error occured when reading the NVM.
0x0010	0x230C	ERC_MEMORY_FAILURE	The HANGINT error occured when erasing a sector of the NVM.
0x0010	0x080C	ERC_MEMORY_FAILURE	The NVM Master was either not idle before reading the NVM or does not get ready after sequencer operation (sector erase, blank check)
0x0010	0x200C	ERC_MEMORY_FAILURE	Flash interface reported that a word could not be written correctly.
0x0010	0x090C	ERC_MEMORY_FAILURE	The NVM Master was not idle before writing to the NVM.

Table 43-35. SHE Used Error Codes

Cancel Error	Command Error	Main Error	Description
0x0010	0x500C	ERC_MEMORY_FAILURE	The sector swap during the NVM update failed. The old values are still active but redundancy is reduced.
0x0010	0x0C0C	ERC_MEMORY_FAILURE	A required slot is not available due to an NVM error during start-up.
0x0010	0x0D0C	ERC_MEMORY_FAILURE	A RAM buffered NVM slot is corrupted by a double bit error.
0x0010	0x0E05	ERC_KEY_EMPTY	A slot needed for authentication, encryption, decryption MAC generation or verfication is empty.
0x0010	0x1107	ERC_KEY_WRITE_PROTECTED	A slot to be updated is write protected.
0x0010	0x4F0C	ERC_MEMORY_FAILURE	The slot update failed and the slot still shows the old value but with reduced redundancy.
0x0010	0x510C	ERC_MEMORY_FAILURE	The slot update failed due to multiple defects and no longer contains any valid key.
0x0010	0x4F01	ERC_NO_ERROR	The slot update was incomplete. The slot contains the new value but the redundancy of the control marker has been reduced.
0x0010	0x140D	ERC_GENERAL_ERROR	TRNG returned an autocorrelation error.
0x0010	0x250D	ERC_GENERAL_ERROR	TRNG returned a CRNGT error.
0x0010	0x260D	ERC_GENERAL_ERROR	The TRNG returned a “Lost Sample” error.
0x0010	0x160D	ERC_GENERAL_ERROR	TRNG did not return RND on time.
0x0010	0x240D	ERC_GENERAL_ERROR	TRNG returned a “Von Neuman” error.

Table 43-36. SHE Used Cancel Codes

Cancel Error	Command Error	Main Error	Description
0x0010	See Table 43-35	See Table 43-35	Command was not cancelled.
0x0000	n.a.	n.a.	Cancellation requested but not yet processed.
0x0001	0x000E	ERC_CANCELED	Cancelled without any error.
0x010D	0x000E	ERC_CANCELED	AES engine could not be stopped during cancellation.
0x020D	0x000E	ERC_CANCELED	Internal register access invoked a bus error during cancellation.

43.5 References

1. SHE Functional Specification v1.1 (rev 439) can be requested at from OEM HIS group
2. Specification for the Advanced Encryption Standard (AES): http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
3. SP 800-38 A -- Recommendation for Block Cipher Modes of Operation: Methods and Techniques: https://csrc.nist.gov/publications/detail/sp/800-38a/final
4. SP 800-38 B -- Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication: http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
5. Menezes, van Oorschot, Vanstone: Handbook of Applied Cryptography(HAC), August 2001, CRC Press, ISBN 0-8493-8523-7, http://www.cacr.math.uwaterloo.ca/hac/

44. Media Local Bus Interface (MediaLB)

This chapter explains the functions and operations of the MediaLB* in FCR4 Cluster.

44.1 Overview of MediaLB

The MediaLB comprises a MediaLB functional block, a AHB Master and Slave interface, and a 9Kbit RAM for the local channel buffer. The MediaLB is an interface that implements the required functionality of a Media Local Bus Device for connecting the device with a MOST (Media Oriented System Transport, which is a multimedia-type in- vehicle LAN standard) network as outlined in reference MediaLB Specification by SMSC (TB0400AN3V0, rev 3.0 issued on February 2006). The MediaLB transfers data between the MediaLB Bus and the system memory, using the AHB Master Interface. In order to operate, MediaLB configuration registers must be set via the AHB Slave Interface. The features of the MediaLB module are listed in this section.

Features of the MediaLB

- Implements the Physical and Link Layer requirements outlined in MediaLB Specification rev 3.0
- Transmits commands and data when functioning as the transmitting device
- Receives data and transmits status responses when functioning as the receiving device
- Detection of the device lock or unlock from the MediaLB Frame.
- Handling of System Channel commands
- Supports 16 logical channels
- Each logical channel can be programmed as synchronous, asynchronous, isochronous and control channel type and as transmit or receive
- Loop back mode between the logical channel 0 (reception) and logical channel 1(transmission)
- Can operate in DMA mode (ping-pong buffering and circular buffering) and in IO mode
- Built-in dual-port RAM as local channel buffer for compensating the bus latency
- Supports 8-bit, 16-bit and 32-bit bus accesses to registers in MediaLB
- Configurable protection for read and write accesses to registers in MediaLB
- Programmable for 256Fs, 512Fs and 1024Fs transfer rates of operation at either 44.0kHz, 48.0kHz, or 48.1kHz.
- 3-pin MediaLB Mode

Abbreviations

This section lists the terms and abbreviations used in this chapter.

Table 44-1. Terms and Abbreviations

Term	Meaning
MediaLB	The function block that implements MediaLB (Media Local Bus), which is a local bus specification, for connecting IC devices with a MOST (Media Oriented System Transport), which is a multimedia-type in- vehicle LAN standard network.
Quadlet	Indicates a 4-byte unit.
Sync data	Data of stream signals, such as audio data.
Async data	Packet data.

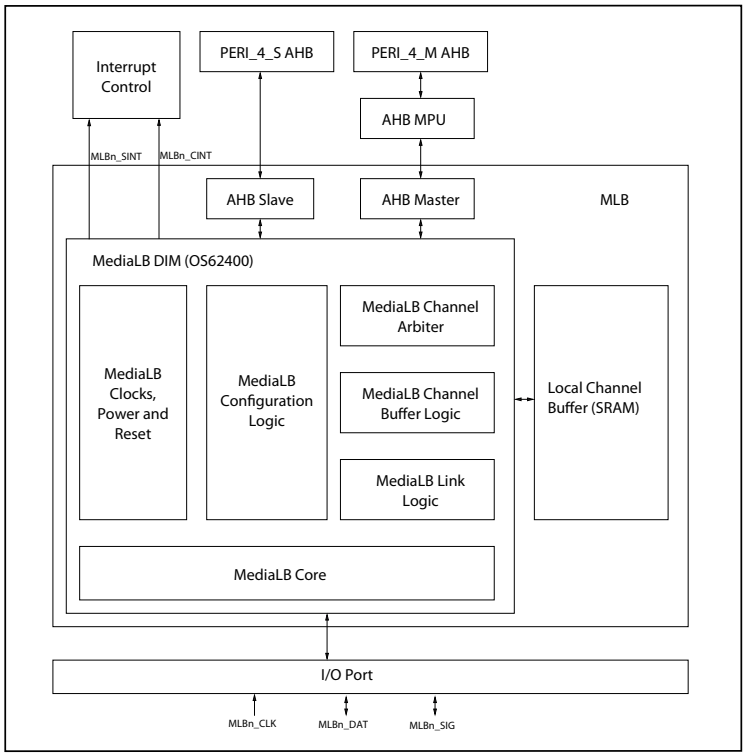
Table 44-1. Terms and Abbreviations

Term	Meaning
Control data	Data of control signals.
Local channel buffer	The data storage area in the MediaLB
IO mode	The mode capable of exchanging transmission/reception data with the MediaLB from CPU.
DMA mode	The mode in which the MediaLB operates as the bus master of the AHB bus. Transmission/reception data is exchanged between the MediaLB and system memory via the AHB bus.
Current buffer	The data area on the AHB bus that is read/written in DMA mode.
Current buffer address	Indicates the address area that is output next on the AHB bus in DMA mode after processing the current buffer area. Set in the channel n next buffer configuration register (MLBn_CNBCRn).
Previous buffer	The data area processed before the current buffer in DMA mode.
Previous buffer address	The address area processed before the current buffer in DMA mode.
Ping-pong buffering	One of two data transfer methods in DMA mode. If the current buffer is filled, an interrupt occurs when moving to the next buffer and so the next address to be used next can be set again by software. Therefore, it is possible to transmit/receive data while changing the transfer destination data storage area one after another.
Circular buffering	One of two data transfer methods in DMA mode. Performs data transfers repeatedly for the data storage area indicated by the current buffer. No interrupt occurs at the completion of a transfer.
n	Indicates a logical channel number (n=0..15)

Block diagram

Figure 44-1 shows the block diagram of MediaLB module.

Figure 44-1. MediaLB Block Diagram



MediaLB DIM

The function block of MediaLB is provided by SMSC IP OS62400 MediaLB Device Interface Macro. It implements the Physical and Link Layer requirements outlined in the MediaLB Specification by SMSC (TB0400AN3V0, rev 3.0 issued on February 2006).

Local Channel Buffer (SRAM)

A 9kbit SRAM is used for buffering of logical channel data for compensating the bus latency. It stores up to 256 quadlets plus tag information (256 words x 36 bits).

1. AHB Master Interface

   The AHB Master Interface allows the MediaLB to access the system memory. It converts the MediaLB Host Bus Interface (HBI) protocol to AHB protocol.

2. AHB Slave Interface

   The AHB Master Interface allows the bus masters to access the MediaLB configuration and status registers. It converts the AHB protocol to MediaLB Peripheral Bus Interface (PBI).

Note:

In devices that use MediaLB, a minimum PERI4 clock value is specified. Please refer to the Internal Clock Timing table in the device-specific datasheet.

In addition, the system must be able to handle 6 MByte/sec on the bus. This means that, depending on the bus traffic as well as access to system RAM by other masters, higher PERI4 clock frequencies are recommended.

MediaLB IO Mode

The IO mode dictates a particular method used for transferring data between the local channel buffer and the system memory. For channels configured to receive data from the MediaLB interface, system software is responsible for periodically unloading RX data from the local channel buffer. For channels configured to transmit data to the MediaLB Interface, system

software is responsible for periodically loading the local channel buffer with TX data. The MLBn_CCBCRn and MLBn_CNBCRn registers, accessed through the AHB Slave interface, are used as a data receive buffer and a data transmit buffer, respectively.

For details on the IO Mode operation, please contact SMSC.

MediaLB DMA Mode

The DMA mode is a mode in which MediaLB accesses the system memory via the AHB Master bus. The system memory contains transmitted and received data, which can be read by software and the MediaLB.

When MediaLB is used in the DMA mode, data is transmitted and received via the system memory. When used in the DMA mode, MediaLB functions as a master of the AHB bus, reading and writing data from and to the system memory. When MediaLB accesses the AHB bus, Buffer Current Address BCA[15:0] in Channel n Current Buffer Configuration Register (MLBn_CCBCRn) for MediaLB is output as the address of the AHB bus. In the DMA mode, two types of buffering is supported: Ping-pong and circular buffering.

In the ping-pong buffering, if the current buffer is filled, an interrupt occurs when moving to the next buffer and so the next address to be used next can be set again by software. Therefore, it is possible to transmit/receive data while changing the transfer destination data storage area one after another.

In the circular buffering, a buffer from a start address to an end address is used in a loop indefinitely by setting the start and end addresses in Next Buffer Configuration Register (MLBn_CNBCRn) until the software sets to “0” the RDY bit in Channel n Status Configuration Register (MLBn_CSCRn). This type of buffering should be used only for synch channels. For details on the DMA Mode operation, please contact SMSC.

Local channel buffer

The local channel buffer comprises RAM of 256 words x 36 bits, where each word contains 4 quadlets plus tag information. The local channel buffer is in MediaLB and used by each logical channel to store transmitted and received data temporarily. To assign a RAM area, Local Channel n Buffer Configuration Register (MLBn_LCBCRn) is used to set the start address (SA bit) and a length (BD bit) of the RAM area. Furthermore, in the IO mode, an interrupt can be triggered to each logical channel by setting a threshold value (TH bit) using Local Channel n Buffer Configuration Register (MLBn_LCBCRn).

For details on the Local channel buffer operation, please contact SMSC.

Interrupts by MediaLB

This section describes interrupts that MediaLB has. MediaLB has two types of interrupts* Channel interrupts indicating the state of each channel buffer (Cint) and System interrupts indicating the state of the MediaLB system (Sint).

A channel interrupt is triggered by the status or error information retained by Channel n Status Configuration Register (MLBn_CSCRn). Channel interrupts are maskable on a channel basis via the MLBn_CECRn register. To know in which channel an interrupt has been triggered, Channel Interrupt Configuration Register (MLBn_CICR) needs to be read. Each bit in Channel Interrupt Configuration Register (MLBn_CICR) is cleared to 0 by clearing the factor that has triggered an interrupt to each channel or masking the interrupt using the mask bit. The root cause of the interrupt can be determined by reading the current and previous status fields in the MLBn_CSCRn register. The following tables describe the interrupt flags for channel interrupts and corresponding interrupt cause factors. The interrupt flag is changed to “1” when an interrupt cause is detected. There are bits that change the meaning of the interrupt flag, depending on the operation mode (IO/DMA mode).

Note: *Data interrupts (mlb_dint[30:0]) provided by SMSC IP OS62400 are not supported. Reason is that data interrupts are provided for customers who want to attach an external DMA Controller directly to OS62400 and load/unload the local channel buffers in IO Mode, which is not FCR4 Cluster case

.

Table 44-2. Interrupt flags and causes common across IO and DMA mode

Interrupt Flag	Register	Interrupt Source	Valid Channel	Interrupt Mask	Interrupt Flag Clear
STS[0]	MLBn_ CSCRn	Current Buffer Protocol Error	All channels with reception setting, and async and control channels with trans-mission setting.	MLBn_CECRn: MASK[0]	Writing “1” to STS[0]
STS[1]		Current Buffer Break	Async and control channels	MLBn_CECRn: MASK[1]	Writing “1” to STS[1]
STS[4]		Buffer Error	Sync (both transmission/reception) and isochronous channels with reception setting.	MLBn_CECRn: MASK[4]	Writing “1” to STS[4]
STS[6]		Frame Sync Lost	Sync channels (both transmission/ reception)	MLBn_CECRn: MASK[6]	Writing “1” to STS[6]

Table 44-3. Interrupt flags and causes valid only in IO mode.

Interrupt Flag	Register	Interrupt Source	Valid Channel	Interrupt Mask	Interrupt Flag Clear
STS[2]	MLBn_ CSCRn	Receive service requests	All channels enabled for receiving.	MLBn_CECRn: MASK[2]	Writing “1” to STS[2]
STS[3]		Transmit service request	All channels enabled for transmitting.	MLBn_CECRn: MASK[3]	Writing “1” to STS[3]
STS[8]		Received packet aborted	Async and control channels enabled for receiving	MLBn_CECEn: MASK[2]	Writing “1” to STS[8]

Table 44-4. Interrupt flags and causes valid in DMA mode only

Interrupt Flag	Register	Interrupt Source	Valid Channel	Interrupt Mask	Interrupt Flag Clear
STS[2]	MLBn_ CSCRn	Current buffer ends	All channels (both transmission/reception)	MLBn_CECRn: MASK[2]	Writing “1” to STS[2]
STS[3]		Current buffer starts	All channels (both transmission/reception)	MLBn_CECRn: MASK[3]	Writing “1” to STS[3]
STS[5]		Host bus error (received AHB bus error response)	All channels (both transmission/reception).	*	Writing “1” to STS[5]
STS[8]		Previous buffer protocol error	All channels enabled for receiving or async and control channel enabled for transmitting	MLBn_CECEn: MASK[2]	Writing “1” to STS[8]
STS[9]		Previous buffer break	All channels (both transmission/reception).	MLBn_CECRn: MASK[1]	Writing “1” to STS[9]
STS[10]		Previous buffer end	All channels (both transmission/reception)	MLBn_CECRn: MASK[2]	Writing “1” to STS[10]
STS[11]		Previous buffer start	All channels (both transmission/reception)	MLBn_CECRn: MASK[3]	Writing “1” to STS[11]

*Cannot be masked. Setting the cause factor flag to “1” triggers an interrupt.

Note:

A system interrupt is triggered when a system command, locking or unlocking is detected, as shown in Table 44-5. An interrupt flag is changed to “1” when an interrupt cause is detected. Each interrupt flag has a corresponding interrupt mask bit. Setting an interrupt mask to “1” enables masking.

Table 44-5. Interrupt flags and interrupt cause factors for system interrupts

Interrupt Source	Register	Interrupt Flag	Interrupt Mask	Interrupt Flag Clear
Detecting a reset (MlbRset(FE$_H$))	MLBn_SSCR	SDR	MLBn_SMCR: SMR	Writing “1” to SDR
Detecting a network lock (MOST_Lock(F0$_H$))		SDNL	MLBn_SMCR: SMNL	Writing “1” to SDNL
Detecting a network unlock (MOST_Unlock(E2$_H$))		SDNU	MLBn_SMCR: SMNU	Writing “1” to SDNU
Detecting a channel scan (MlbScan(E4$_H$))		SDCS	MLBn_SMCR: SMCS	Writing “1” to SDCS
Detecting a sub-command (MlbSubCmd(E6$_H$))		SDSC	MLBn_SMCR: SMSC	Writing “1” to SDSC
Detecting a MediaLB lock		SDML	MLBn_SMCR: SMML	Writing “1” to SDML
Detecting a MediaLB unlock		SDMU	MLBn_SMCR: SMMU	Writing “1” to SDMU

Loop Back Mode

To facilitate debugging of transmission paths, MediaLB supports the loop back test mode.

Setting the LBM bit in Device Control Configuration Register (MLBn_DCCR) to “1” causes data to be received via Channel 0 and transmitted back via Channel 1.To use the loop back test mode, use the following steps to set the mode*:

3. Set the logical channel addresses for Channels 0 and 1. (Same address cannot be used for different channels.)
4. Set Channel 0 to any channel type. Enable “Receipt” and set “Enable channel”.
5. Set Channel 1 to the same channel type as that of Channel 0, enable “Transmit” and set “Enable channel”.
6. Set the loop back mode bit (MLBn_DCCR:LBM=”1”).

Note:

*During the loop back mode, a protocol error or a break is prohibited for both Channels 0 and 1. During the loop back mode, the Next Buffer Ready bits for Channels 0 and 1 remains cleared (MLBn_CSCR0:RDY=MLBn_CSCR1:RDY=”0”). During the loop back mode, little-endian mode must be disabled (MLBn_DCCR=MLE=”0”). During the loop back mode, ishochronous packet lengths must be quadlet multiples.

44.2 MediaLB Register Set

The MediaLB module contains various registers to configure its operation, to monitor its status and, when configured in IO Mode, to access the receive and transmit data buffers. The MediaLB module is allocated 1KB of MCU address space for mapping the Configuration and Status Registers (i.e. CSRs). The address area allocated to MediaLB and the Control and Status Registers in MediaLB are explained in this section.

The suffix ‘n’ in the register name indicates that the register is an instance ‘n’ of the module.

Registers of MediaLB

The following registers are available for each instance of MediaLB:

- MediaLBn Device Control Configuration Register (MLBn_DCCR)
- MediaLBn System Status Configuration Register (MLBn_SSCR)
- MediaLBn System Data Configuration Register (MLBn_SDCR)
- MediaLBn System Mask Configuration Register (MLBn_SMCR)
- MediaLBn Version Control Configuration Register (MLBn_VCCR)
- MediaLBn Synchronous Base Address Configuration Register (MLBn_SBCR)
- MediaLBn Asynchronous Base Address Configuration Register (MLBn_ABCR)
- MediaLBn Control Base Address Configuration Register (MLBn_CBCR)
- MediaLBn Isochronous Base Address Configuration Register (MLBn_IBCR)
- MediaLBn Channel Interrupt Configuration Register (MLBn_CICR)
- MediaLBn AHB Master Control Register (MLBn_AHBMCTL)
- MediaLBn Channel Entry Configuration Register (MLBn_CECR0 - MLBn_CECR15)
- MediaLBn Channel Status Configuration Register (MLBn_CSCR0 - MLBn_CSCR15)
- MediaLBn Channel Current Buffer Configuration Register (MLBn_CCBCR0 - MLBn_CCBCR15)
- MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0 - MLBn_CNBCR15)
- MediaLBn Local Channel Buffer Configuration Register (MLBn_LCBCR0 - MLBn_LCBCR15)
- MediaLBn Module Identification Register (MLBn_MID)

Arrangement of MediaLB Registers in Memory

Table 44-6. MediaLB Register Map

Offset	+3	+2	+1	+0
0x00000000	MLBn_DCCR 00000000 00000000 00000000 00000000			
0x00000004	MLBn_SSCR 00000000 00000000 00000000 00000000			
0x00000008	MLBn_SDCR 00000000 00000000 00000000 00000000			
0x0000000C	MLBn_SMCR 00000000 00000000 00000000 01100000			
0x00000010 - 0x00000018	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000001C	MLBn_VCCR 00000000 00000000 00000011 00000000			

Table 44-6. MediaLB Register Map

Offset	+3	+2	+1	+0
0x00000020	MLBn_SBCR 00000000 00000000 00000000 00000000			
0x00000024	MLBn_ABCR 00000000 00000000 00000000 00000000			
0x00000028	MLBn_CBCR 00000000 00000000 00000000 00000000			
0x0000002C	MLBn_IBCR 00000000 00000000 00000000 00000000			
0x00000030	MLBn_CICR 00000000 00000000 00000000 00000000			
0x00000034 - 0x00000038	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x0000003C	MLBn_AHBMCTL 00000000 00000000 00000000 00000000			
0x00000040	MLBn_CECR0 00000000 00000000 00000000 00000000			
0x00000044	MLBn_CSCR0 10000000 00000000 00000000 00000000			
0x00000048	MLBn_CCBCR0 00000000 00000000 00000000 00000000			
0x0000004C	MLBn_CNBCR0 00000000 00000000 00000000 00000000			
0x00000050	MLBn_CECR1 00000000 00000000 00000000 00000000			
0x00000054	MLBn_CSCR1 10000000 00000000 00000000 00000000			
0x00000058	MLBn_CCBCR1 00000000 00000000 00000000 00000000			
0x0000005C	MLBn_CNBCR1 00000000 00000000 00000000 00000000			
0x00000060	MLBn_CECR2 00000000 00000000 00000000 00000000			
0x00000064	MLBn_CSCR2 10000000 00000000 00000000 00000000			
0x00000068	MLBn_CCBCR2 00000000 00000000 00000000 00000000			
0x0000006C	MLBn_CNBCR2 00000000 00000000 00000000 00000000			

Table 44-6. MediaLB Register Map

Offset	+3	+2	+1	+0
0x00000070	MLBn_CECR3 00000000 00000000 00000000 00000000			
0x00000074	MLBn_CSCR3 10000000 00000000 00000000 00000000			
0x00000078	MLBn_CCBCR3 00000000 00000000 00000000 00000000			
0x0000007C	MLBn_CNBCR3 00000000 00000000 00000000 00000000			
0x00000080	MLBn_CECR4 00000000 00000000 00000000 00000000			
0x00000084	MLBn_CSCR4 10000000 00000000 00000000 00000000			
0x00000088	MLBn_CCBCR4 00000000 00000000 00000000 00000000			
0x0000008C	MLBn_CNBCR4 00000000 00000000 00000000 00000000			
0x00000090	MLBn_CECR5 00000000 00000000 00000000 00000000			
0x00000094	MLBn_CSCR5 10000000 00000000 00000000 00000000			
0x00000098	MLBn_CCBCR5 00000000 00000000 00000000 00000000			
0x0000009C	MLBn_CNBCR5 00000000 00000000 00000000 00000000			
0x000000A0	MLBn_CECR6 00000000 00000000 00000000 00000000			
0x000000A4	MLBn_CSCR6 10000000 00000000 00000000 00000000			
0x000000A8	MLBn_CCBCR6 00000000 00000000 00000000 00000000			
0x000000AC	MLBn_CNBCR6 00000000 00000000 00000000 00000000			
0x000000B0	MLBn_CECR7 00000000 00000000 00000000 00000000			
0x000000B4	MLBn_CSCR7 10000000 00000000 00000000 00000000			
0x000000B8	MLBn_CCBCR7 00000000 00000000 00000000 00000000			

Table 44-6. MediaLB Register Map

Offset	+3	+2	+1	+0
0x000000BC	MLBn_CNBCR7 00000000 00000000 00000000 00000000			
0x000000C0	MLBn_CECR8 00000000 00000000 00000000 00000000			
0x000000C4	MLBn_CSCR8 10000000 00000000 00000000 00000000			
0x000000C8	MLBn_CCBCR8 00000000 00000000 00000000 00000000			
0x000000CC	MLBn_CNBCR8 00000000 00000000 00000000 00000000			
0x000000D0	MLBn_CECR9 00000000 00000000 00000000 00000000			
0x000000D4	MLBn_CSCR9 10000000 00000000 00000000 00000000			
0x000000D8	MLBn_CCBCR9 00000000 00000000 00000000 00000000			
0x000000DC	MLBn_CNBCR9 00000000 00000000 00000000 00000000			
0x000000E0	MLBn_CECR10 00000000 00000000 00000000 00000000			
0x000000E4	MLBn_CSCR10 10000000 00000000 00000000 00000000			
0x000000E8	MLBn_CCBCR10 00000000 00000000 00000000 00000000			
0x000000EC	MLBn_CNBCR10 00000000 00000000 00000000 00000000			
0x000000F0	MLBn_CECR11 00000000 00000000 00000000 00000000			
0x000000F4	MLBn_CSCR11 10000000 00000000 00000000 00000000			
0x000000F8	MLBn_CCBCR11 00000000 00000000 00000000 00000000			
0x000000FC	MLBn_CNBCR11 00000000 00000000 00000000 00000000			
0x00000100	MLBn_CECR12 00000000 00000000 00000000 00000000			
0x00000104	MLBn_CSCR12 10000000 00000000 00000000 00000000			

Table 44-6. MediaLB Register Map

Offset	+3	+2	+1	+0
0x00000108	MLBn_CCBCR12 00000000 00000000 00000000 00000000			
0x0000010C	MLBn_CNBCR12 00000000 00000000 00000000 00000000			
0x00000110	MLBn_CECR13 00000000 00000000 00000000 00000000			
0x00000114	MLBn_CSCR13 10000000 00000000 00000000 00000000			
0x00000118	MLBn_CCBCR13 00000000 00000000 00000000 00000000			
0x0000011C	MLBn_CNBCR13 00000000 00000000 00000000 00000000			
0x00000120	MLBn_CECR14 00000000 00000000 00000000 00000000			
0x00000124	MLBn_CSCR14 10000000 00000000 00000000 00000000			
0x00000128	MLBn_CCBCR14 00000000 00000000 00000000 00000000			
0x0000012C	MLBn_CNBCR14 00000000 00000000 00000000 00000000			
0x00000130	MLBn_CECR15 00000000 00000000 00000000 00000000			
0x00000134	MLBn_CSCR15 10000000 00000000 00000000 00000000			
0x00000138	MLBn_CCBCR15 00000000 00000000 00000000 00000000			
0x0000013C	MLBn_CNBCR15 00000000 00000000 00000000 00000000			
0x00000140 - 0x0000027C	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x00000280	MLBn_LCBCR0 00000000 01000000 00000000 00000000			
0x00000284	MLBn_LCBCR1 00000000 01000000 00000000 00000000			
0x00000288	MLBn_LCBCR2 00000000 01000000 00000000 00000000			
0x0000028C	MLBn_LCBCR3 00000000 01000000 00000000 00000000			

Table 44-6. MediaLB Register Map

Offset	+3	+2	+1	+0
0x00000290	MLBn_LCBCR4 00000000 01000000 00000000 00000000			
0x00000294	MLBn_LCBCR5 00000000 01000000 00000000 00000000			
0x00000298	MLBn_LCBCR6 00000000 01000000 00000000 00000000			
0x0000029C	MLBn_LCBCR7 00000000 01000000 00000000 00000000			
0x000002A0	MLBn_LCBCR8 00000000 01000000 00000000 00000000			
0x000002A4	MLBn_LCBCR9 00000000 01000000 00000000 00000000			
0x000002A8	MLBn_LCBCR10 00000000 01000000 00000000 00000000			
0x000002AC	MLBn_LCBCR11 00000000 01000000 00000000 00000000			
0x000002B0	MLBn_LCBCR12 00000000 01000000 00000000 00000000			
0x000002B4	MLBn_LCBCR13 00000000 01000000 00000000 00000000			
0x000002B8	MLBn_LCBCR14 00000000 01000000 00000000 00000000			
0x000002BC	MLBn_LCBCR15 00000000 01000000 00000000 00000000			
0x000002C0 - 0x000002F8	reserved XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX			
0x000002FC	MLBn_MID 00000000 00000000 00000000 00000000			

44.2.1 MediaLBn Device Control Configuration Register (MLBn_DCCR)

The MediaLBn Device Control Configuration Register (MLBn_DCCR) is used to control basic features of the MediaLB, such as clock rate, pinout, lock status, enable and device addressing.

MediaLBn Device Control Configuration Register (MLBn_DCCR)

Figure 44-2. MediaLBn Device Control Configuration Register (MLBn_DCCR)

MLBn_DCCR																															
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**
MDE	LBM	MCS[1]	MCS[0]	M5PS	MLK	MLE	MHRE	MRS	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	MDA[7]	MDA[6]	MDA[5]	MDA[4]	MDA[3]	MDA[2]	MDA[1]	MDA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	Rp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-7. MediaLBn Device Control Configuration Register (MLBn_DCCR) bits

Bit Position	Bit Field Name	Bit Description
[31]	MDE	MediaLB Device Enable Enables the operation of MediaLB interface '0': Disables MediaLB Interface. '1': Enables MediaLB Interface based on the other bits in the register.
[30]	LBM	Loop-Back Mode Enable Sets whether the loopback test for the MediaLB bus between even logical channel N (reception setting) and odd logical channel N+1 (transmission setting) is enabled or disabled. '0': Normal operation. Disables Loop-back test mode. '1': Enables Loop-back test mode. Note - For more information on Loop-BackTest mode please contact SMSC support.
[29:28]	MCS	MediaLB Clock Select Sets the MediaLB transfer rate. This field must be programmed by the system software to reflect the MLBn_MLBCLK speed. '00': 256Fs - supports 8 quadlets per frame '01': 512Fs - supports 16 quadlets per frame '10': 1024Fs - supports 32 quadlets per frame '11': Reserved It is prohibited to set this bit field to '11'

Table 44-7. MediaLBn Device Control Configuration Register (MLBn_DCCR) bits

Bit Position	Bit Field Name	Bit Description
[27]	M5PS	MediaLB 5-Pin Select Selects MediaLB 5-pin or 3-pin configuration '0': 3-pin MediaLB mode '1': 5-pin MediaLB mode **Note:** This device does not support 5-pin MediaLB interface and therefore writing '1' to this bit is prohibited. Make sure the bit is set to '0'
[26]	MLK	MediaLB Lock Indicates whether MediaLB is in sync with the MediaLB frame (lock) '0': Unlocked state. '1': Locked state. When set, it indicates that the MediaLB Port is synchronized to the incoming MediaLB frame. If MLK is clear (unlocked), MLK is set after FRAMESYNC is detected at the same position for three consecutive frames. If MLK is set (locked), MLK is cleared after not receiving FRAMESYNC at the expected time for two consecutive frames. While MLK is set, FRAMESYNC patterns occurring at locations other than the expected one are ignored. **Note:** When MLBn_DCCR:MRS is set to '1', this bit is cleared to '0'. When MLBn_DCCR:MDE is '0', the lock is not detected
[25]	MLE	MediaLB Little Endian mode This field determines how MediaLB data (quadlet-based) is stored in system memory. '0': Big Endian mode '1': Little Endian mode
[24]	MHRE	MediaLB Hardware Reset Enable This bit enables the hardware to automatically reset the MediaLB physical and link layer logic upon reception of the reset command. Resetting is preformed by the reset request from the Intelligent Network Interface Controller (INIC). Resetting is performed when either the global (MLBn_SDCR:MSD = 0x00: all of MediaLB are targeted for reset) MlbReset (0xFE) or device-specific (MLBn_SDCR:MSD = DA: MediaLB designated by Device Address is targeted for reset) MlbReset(0xFE) is received from INIC. '0': Disables resetting upon reception of reset command '1': Enables resetting upon reception of reset command

Table 44-7. MediaLBn Device Control Configuration Register (MLBn_DCCR) bits

Bit Position	Bit Field Name	Bit Description
[23]	MRS	MediaLB Software Reset When set, this bit resets the MediaLB physical and link layer. It is cleared automatically by hardware after the reset execution is complete. '0': Normal operation '1': Reset **Note:** Even when the MLBn_DCCR:MHRE bit is '1' and the reset operation by the receipt of the reset command is in progress, this bit is reset to '0' after the execution of the reset command.
[22:16]	read0	-
[15:8]	read0	-
[7:0]	MDA	MediaLB Device Address The MediaLB Device Address (MDA[8:1]) sets a unique Device Address (DA) for the MediaLB Device. The device address is a 16-bit address allocated to identify the MediaLB device. The DA is used by the system channel MlbScan and MlbReset commands. The received DA (DA[15:0]) operates as the command target when DA[15:9] and DA[0] are '0' and DA[8:1] matches this bit field.

44.2.2 MediaLBn System Status Configuration Register (MLBn_SSCR)

The MediaLBn System Status Configuration Register (MLBn_SSCR) is a register to indicate the status of the MediaLB network. MLBn_SSCR is updated for each MediaLB frame.

MediaLBn System Status Configuration Register (MLBn_SSCR)

Figure 44-3. MediaLBn System Status Configuration Register (MLBn_SSCR)

MLBn_SSCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SSRE	SDMU	SDML	SDSC	SDCS	SDNU	SDNL	SDR
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-8. MediaLBn System Status Configuration Register (MLBn_SSCR)

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	SSRE	System service request enable '0': System service request response disabled '1': System service request response enable System software can set this bit to indicate that the MediaLB Device is present and needs service. When this bit is set to '1', RxStatus (RxStatus (DeviceServiceRequest(0x82)) is transmitted as a response to the MlbScan (0xE4) system command. When RxStatus is transmitted, hardware clears this bit to '0'.
[6]	SDMU	System Detects MediaLB Unlock '0': MediaLB unlock not detected '1': MediaLB unlock detected This bit is set to indicate that the MediaLB Device has unlocked from the MediaLB frame. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMMU), a system interrupt is generated when MediaLB unlock is detected. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software. **Note:** During the locked state, this bit is set to '1' when the MLBn_DCCR:MRS bit is set to '1'.

Table 44-8. MediaLBn System Status Configuration Register (MLBn_SSCR)

Bit Position	Bit Field Name	Bit Description
[5]	SDML	System Detects MediaLB Lock '0': MediaLB lock not detected '1': MediaLB lock detected This bit is set to indicate that the MediaLB Device has locked to the MediaLB frame. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMML), a system interrupt is generated when the MediaLB lock is detected. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[4]	SDSC	System Detects Subcommand '0': Subcommand not detected '1': Subcommand detected This bit is set to indicate that the MediaLB Device has received the MlbSubCmd (0xE6) System Command. The user-defined software command is stored in the MLBn_SDCR register. The decoding of this command is left up to software. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMSC), a system interrupt is generated on detecting MlbSubCmd. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[3]	SDCS	System Detects Channel Scan '0': Channel scan is not detected '1': Channel scan is detected This bit is set to indicate that the MediaLB Device has received the MlbScan (0xE4) System Command. The target DeviceAddress is stored in the MLBn_SDCR register. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMCS), a system interrupt is generated on detection of the MlbScan command. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.

Table 44-8. MediaLBn System Status Configuration Register (MLBn_SSCR)

Bit Position	Bit Field Name	Bit Description
[2]	SDNU	System Detects Network Unlock '0': Network unlock not detected '1': Network unlock detected This bit is set to indicate that the MediaLB Device has received the MOST_Unlock (0xE2) System Command. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMNU), a system interrupt is generated on detection of the MOST_Unlock command. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[1]	SDNL	System Detects Network Lock '0': Network lock not detected '1': Network lock detected This bit is set to indicate that the MediaLB Device has received the MOST_Lock (0xE0) System Command. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMNL), a system interrupt is generated on detection of the MOST_Lock command. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[0]	SDR	System Detects Reset '0': Reset not detected '1': Reset detected This bit is set to indicate that the MediaLB Device has received the MlbReset (0xFE) System Command. The target DeviceAddress is stored in the MLBn_SDCR register. If not masked by the System Mask Configuration Register (MLBn_SMCR:SMR), a system interrupt is generated on detection of the MlbReset command. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit is sticky until cleared by software.

Note:

The MediaLBn System Status Configuration register (MLBn_SSCR) allows system software to monitor and control the status of the MediaLB network. MLBn_SSCR is updated once per frame by hardware during the MediaLB System Channel. Except for the bits associated with MediaLB lock and unlock (MLBn_SSCR.SDMU and MLBn_SSCR.SDML), the bits of the MLBn_SSCR register are not valid until the device is locked to the MediaLB interface. System software must service status events before the start of the next MediaLB frame to prevent the current frame status from being lost.

44.2.3 MediaLBn System Data Configuration Register (MLBn_SDCR)

MediaLBn System Data Configuration Register (MLBn_SDCR) is a register to receive the system channel data for the MediaLB frame. SDCR is updated for each MediaLB frame. System software must read MLBn_SDCR before the start of the next MediaLB frame to prevent the current frame data from being lost.

MediaLBn System Data Configuration Register (MLBn_SDCR)

Figure 44-4. MediaLBn System Data Configuration Register (MLBn_SDCR)

MLBn_SDCR																															
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**
MSD[31]	MSD[30]	MSD[29]	MSD[28]	MSD[27]	MSD[26]	MSD[25]	MSD[24]	MSD[23]	MSD[22]	MSD[21]	MSD[20]	MSD[19]	MSD[18]	MSD[17]	MSD[16]	MSD[15]	MSD[14]	MSD[13]	MSD[12]	MSD[11]	MSD[10]	MSD[9]	MSD[8]	MSD[7]	MSD[6]	MSD[5]	MSD[4]	MSD[3]	MSD[2]	MSD[1]	MSD[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-9. MediaLBn System Data Configuration Register (MLBn_SDCR) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MSD	MediaLB System Data This register is loaded with the data from MLBn_MLBDAT during the System Channel quadlet. The System Data Configuration register (MLBn_SDCR) allows system software to receive control information from the MediaLB Controller. MLBn_SDCR is updated once per frame by hardware during the MediaLB System Channel. System software must read MLBn_SDCR before the start of the next MediaLB frame to prevent the current frame data from being lost.

44.2.4 MediaLBn System Mask Configuration Register (MLBn_SMCR)

MediaLBn System Mask Configuration Register (MLBn_SMCR) sets the system interrupt mask

MediaLB System Mask Configuration Register (MLBn_SMCR)

Figure 44-5. MediaLBn System Mask Configuration Register (MLBn_SMCR)

MLBn_SMCR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	SMMU	SMML	SMSC	SMCS	SMNU	SMNL	SMR
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 44-10. MediaLBn System Mask Configuration Register (MLBn_SMCR) bits

Bit Position	Bit Field Name	Bit Description
[31:8]	read0	-
[7]	read0	-
[6]	SMMU	System Masks MediaLB Unlock Sets whether system interrupt should be masked when unlock was detected from the MediaLB frame. Detection of unlock from the MediaLB frame is indicated by MLBn_SSCR:SDMU. '0': Interrupt due to detection of unlock from the MediaLB frame is not masked. '1': Interrupt due to detection of unlock from the MediaLB frame is masked.
[5]	SMML	System Masks MediaLB Lock Sets whether the system interrupt should be masked when the lock was detected from the MediaLB frame. Detection of the lock from the MediaLB frame is indicated by MLBn_SSCR:SDML. '0': Interrupt due to detection of lock with MediaLB frame is not masked. '1': Interrupt due to detection of lock with MediaLB frame is masked.

Table 44-10. MediaLBn System Mask Configuration Register (MLBn_SMCR) bits

Bit Position	Bit Field Name	Bit Description
[4]	SMSC	System Masks Subcommand Sets whether system interrupt should be masked when the sub-command of the system command, MlbSubCmd(0xE6) is received. The receipt of a subcommand of the system command, MlbSubCmd(0xE6) is indicated by MLBn_SSCR:SDSC. '0': Interrupt due to the receipt of the MlbSubCmd(0xE6) system command is not masked. '1': Interrupt due to the receipt of the MlbSubCmd(0xE6) system command is masked.
[3]	SMCS	System Masks Channel Scan Sets whether system interrupt should be masked when the system command, MlbScan(0xE4) is received. The receipt of system command, MlbScan(0xE4) is indicated by MLBn_SSCR:SDCS. '0': Interrupt due to the receipt of the MlbScan(0xE4) system command is not masked. '1': Interrupt due to the receipt of the MlbScan(0xE4) system command is masked.
[2]	SMNU	System Masks Network Unlock Sets whether system interrupt should be masked when the system command, MOST_Unlock(0xE2) is received. The receipt of the system command, MOST_Unlock(0xE2) is indicated by MLBn_SSCR:SDNU. '0': Interrupt due to the receipt of the MOST_Unlock(0xE2) system command is not masked. '1': Interrupt due to the receipt of the MOST_Unlock(0xE2) system command is masked.
[1]	SMNL	System Masks Network Lock Sets whether system interrupt should be masked when the system command, MOST_Lock(0xE0) is received. The receipt of system command, MOST_Lock(0xE0) is indicated by MLBn_SSCR:SDNL. '0': Interrupt due to the receipt of the MOST_Lock(0xE0) system command is not masked. '1': Interrupt due to the receipt of the MOST_Lock(0xE0) system command is masked.

Table 44-10. MediaLBn System Mask Configuration Register (MLBn_SMCR) bits

Bit Position	Bit Field Name	Bit Description
[0]	SMR	System Masks Reset Sets whether system interrupt should be masked when the system command, MlbReset(0xFE) is received. The receipt of system command, MlbReset(0xFE)is indicated by MLBn_SSCR:SDR. '0': Interrupt due to the receipt of the MlbReset(0xFE) system command is not masked. '1': Interrupt due to the receipt of the MlbReset(0xFE) system command is masked.

44.2.5 MediaLBn Version Control Configuration Register (MLBn_VCCR)

MediaLBn Version Control Configuration Register (MLBn_VCCR) is a register to indicate the MediaLB device version.

MediaLB Version Control Configuration Register (MLBn_VCCR)

Figure 44-6. MediaLBn Version Control Configuration Register (MLBn_VCCR)

MLBn_VCCR																															
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**
UMA[7]	UMA[6]	UMA[5]	UMA[4]	UMA[3]	UMA[2]	UMA[1]	UMA[0]	UMI[7]	UMI[6]	UMI[5]	UMI[4]	UMI[3]	UMI[2]	UMI[1]	UMI[0]	MMA[7]	MMA[6]	MMA[5]	MMA[4]	MMA[3]	MMA[2]	MMA[1]	MMA[0]	MMI[7]	MMI[6]	MMI[5]	MMI[4]	MMI[3]	MMI[2]	MMI[1]	MMI[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Table 44-11. MediaLBn Version Control Configuration Register (MLBn_VCCR) bits

Bit Position	Bit Field Name	Bit Description
[31:24]	UMA	User Major Revision Code User Major Revision Code. For FCR4, this register is not used for revision code. Please refer to MLBn_MID register for the particular version of MediaLB used in the device. Read value of this bit field is "0x00".
[23:16]	UMI	User Minor Revision Code User Minor Revision Code. For FCR4, this register is not used for revision code. Please refer to MLBn_MID register for the particular version of MediaLB used in the device. Read value of this bit field is "0x00".
[15:8]	MMA	MediaLB Major Revision Code MediaLB Major Revision Code. This field identifies the major revision of the MediaLB module (OS62400) on this device. This value is hard-coded by the vendor. **Note:** For OS62400 version code information, refer to the device datasheet.
[7:0]	MMI	MediaLB Minor Revision Code MediaLB Minor Revision Code. This field identifies the minor revision of the MediaLB module (OS62400) implemented on this device. This value is hard-coded by the vendor. **Note:** For OS62400 version code information, refer to the device datasheet.

44.2.6 MediaLBn Synchronous Base Address Configuration Register (MLBn_SBCR)

MediaLBn Synchronous Base Address Configuration Register (MLBn_SBCR) defines the base address for synchronous receive/transmit system memory buffers.

MediaLB Synchronous Base Address Configuration Register (MLBn_SBCR)

Figure 44-7. MediaLBn Synchronous Base Address Configuration Register (MLBn_SBCR)

MLBn_SBCR																															
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**
SRBA[15]	SRBA[14]	SRBA[13]	SRBA[12]	SRBA[11]	SRBA[10]	SRBA[9]	SRBA[8]	SRBA[7]	SRBA[6]	SRBA[5]	SRBA[4]	SRBA[3]	SRBA[2]	SRBA[1]	SRBA[0]	STBA[15]	STBA[14]	STBA[13]	STBA[12]	STBA[11]	STBA[10]	STBA[9]	STBA[8]	STBA[7]	STBA[6]	STBA[5]	STBA[4]	STBA[3]	STBA[2]	STBA[1]	STBA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-12. MediaLBn Synchronous Base Address Configuration Register (MLBn_SBCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	SRBA	Upper Half of Synchronous Receive Base Address for DMA mode These bits allow the system software to define the base address (SRBA[31:16]) for synchronous receive system memory buffers. This base address is shared by all synchronous receive channels and defines the upper 16 bits of the 32-bit address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used
[15:0]	STBA	Upper Half of Synchronous Transmit Base Address for DMA mode These bits allow the system software to define the base address (STBA[31:16]) for synchronous transmit system memory buffers in DMA mode. This base address is shared by all synchronous transmit channels and defines the upper 16 bits of the 32-bit address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used

44.2.7 MediaLBn Asynchronous Base Address Configuration Register (MLBn_ABCR)

MediaLBn Asynchronous Base Address Configuration Register (MLBn_ABCR) defines the base address for asynchronous receive/transmit system memory buffers.

MediaLB Asynchronous Base Address Configuration Register (MLBn_ABCR)

Figure 44-8. MediaLBn Asynchronous Base Address Configuration Register (MLBn_ABCR)

MLBn_ABCR																															
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**
ARBA[15]	ARBA[14]	ARBA[13]	ARBA[12]	ARBA[11]	ARBA[10]	ARBA[9]	ARBA[8]	ARBA[7]	ARBA[6]	ARBA[5]	ARBA[4]	ARBA[3]	ARBA[2]	ARBA[1]	ARBA[0]	ATBA[15]	ATBA[14]	ATBA[13]	ATBA[12]	ATBA[11]	ATBA[10]	ATBA[9]	ATBA[8]	ATBA[7]	ATBA[6]	ATBA[5]	ATBA[4]	ATBA[3]	ATBA[2]	ATBA[1]	ATBA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-13. MediaLBn Asynchronous Base Address Configuration Register (MLBn_ABCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	ARBA	Upper Half of Asynchronous Receive Base Address for DMA mode These bits allow system software to define the base address (ARBA[31:16]) for asynchronous receive system memory buffers. This base address is shared by all asynchronous receive channels and defines the upper 16 bits of the 32-bit system bus address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used
[15:0]	ATBA	Upper Half of Asynchronous Transmit Base Address for DMA mode These bits allow the system software to define the base address (ATBA[31:16]) for asynchronous transmit system memory buffers in DMA mode. This base address is shared by all asynchronous transmit channels and defines the upper 16 bits of the 32-bit system bus address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used

44.2.8 MediaLBn Control Base Address Configuration Register (MLBn_CBCR)

MediaLBn Control Base Address Configuration Register (MLBn_CBCR) defines the base address for control receive/transmit system memory buffers.

MediaLB Control Base Address Configuration Register (MLBn_CBCR)

Figure 44-9. MediaLBn Control Base Address Configuration Register (MLBn_CBCR)

MLBn_CBCR																															
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**
CRBA[15]	CRBA[14]	CRBA[13]	CRBA[12]	CRBA[11]	CRBA[10]	CRBA[9]	CRBA[8]	CRBA[7]	CRBA[6]	CRBA[5]	CRBA[4]	CRBA[3]	CRBA[2]	CRBA[1]	CRBA[0]	CTBA[15]	CTBA[14]	CTBA[13]	CTBA[12]	CTBA[11]	CTBA[10]	CTBA[9]	CTBA[8]	CTBA[7]	CTBA[6]	CTBA[5]	CTBA[4]	CTBA[3]	CTBA[2]	CTBA[1]	CTBA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-14. MediaLBn Control Base Address Configuration Register (MLBn_CBCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	CRBA	Upper Half of Control Receive Base Address for DMA mode These bits allow the system software to define the base address (CRBA[31:16]) for control receive system memory buffers. This base address is shared by all control receive channels and defines the upper 16 bits of the 32-bit system bus address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used
[15:0]	CTBA	Upper Half of Control Transmit Base Address for DMA mode These bits allow the system software to define the base address (CTBA[31:16]) for control transmit system memory buffers in DMA mode. This base address is shared by all control transmit channels and defines the upper 16 bits of the 32-bit system bus address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used

44.2.9 MediaLBn Isochronous Base Address Configuration Register (MLBn_IBCR)

MediaLBn Isochronous Base Address Configuration Register (MLBn_IBCR) defines the base address for control receive/ transmit system memory buffers.

MediaLB Isochronous Base Address Configuration Register (MLBn_IBCR)

Figure 44-10. MediaLBn Isochronous Base Address Configuration Register (MLBn_IBCR)

MLBn_IBCR																															
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**
IRBA[15]	IRBA[14]	IRBA[13]	IRBA[12]	IRBA[11]	IRBA[10]	IRBA[9]	IRBA[8]	IRBA[7]	IRBA[6]	IRBA[5]	IRBA[4]	IRBA[3]	IRBA[2]	IRBA[1]	IRBA[0]	ITBA[15]	ITBA[14]	ITBA[13]	ITBA[12]	ITBA[11]	ITBA[10]	ITBA[9]	ITBA[8]	ITBA[7]	ITBA[6]	ITBA[5]	ITBA[4]	ITBA[3]	ITBA[2]	ITBA[1]	ITBA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-15. MediaLBn Isochronous Base Address Configuration Register (MLBn_IBCR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	IRBA	Upper Half of Isochronous Receive Base Address for DMA mode These bits allow the system software to define the base address (IRBA[31:16]) for isochronous receive system memory buffers. This base address is shared by all isochronous receive channels and defines the upper 16 bits of the 32-bit system bus address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used
[15:0]	ITBA	Upper Half of Isochronous Transmit Base Address for DMA mode These bits allow the system software to define the base address (ITBA[31:16]) for isochronous transmit system memory buffers in DMA mode. This base address is shared by all isochronous transmit channels and defines the upper 16 bits of the 32-bit system bus address for these channels. This bit field is only used in DMA mode. In IO mode these bits are not used

44.2.10 MediaLBn Channel Interrupt Configuration Register (MLBn_CICR)

MediaLBn Channel Interrupt Configuration Register (MLBn_CICR) reflects the channel interrupt status. These bits are set by hardware when a channel interrupt is generated.

MediaLB Channel Interrupt Configuration Register (MLBn_CICR)

Figure 44-11. MediaLBn Channel Interrupt Configuration Register (MLBn_CICR)

MLBn_CICR																															
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**
read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	CNSU[15]	CNSU[14]	CNSU[13]	CNSU[12]	CNSU[11]	CNSU[10]	CNSU[9]	CNSU[8]	CNSU[7]	CNSU[6]	CNSU[5]	CNSU[4]	CNSU[3]	CNSU[2]	CNSU[1]	CNSU[0]
Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-16. MediaLBn Channel Interrupt Configuration Register (MLBn_CICR) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	read0	-
[15:0]	CNSU	Channel Status Update (for Channel 15 through 0) CNSU[n] indicates whether there is an interrupt for Channel n. '0': If CNSU[n] is '0' there is no interrupt for Channel n. '1': If CNSU[n] is '1' Channel n has an interrupt. Note - Writing to the MLBn_CICR register has no effect. To clear a particular bit in MLBn_CICR, software must clear all of the unmasked status bits in the corresponding MLBn_CSCRn register. For example, if MLBn_CNSU[4] is set, writing FFFFh to MLBn_CSCR4[15:0] releases the channel interrupt (MLBn_CINT) and clears MLBn_CICR[4].

44.2.11 MediaLBn AHB Master Control Register (MLBn_AHBMCTL)

MediaLBn AHB Master Control Register (MLBn_AHBMCTL) controls the AHB Bus Request from MediaLB Master.

MediaLB AHB Master Control Register (MLBn_AHBMCTL)

Figure 44-12. MediaLBn AHB Master Control Register (MLBn_AHBMCTL)

MLBn_AHBMCTL																															
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**
MAXTRANS[15]	MAXTRANS[14]	MAXTRANS[13]	MAXTRANS[12]	MAXTRANS[11]	MAXTRANS[10]	MAXTRANS[9]	MAXTRANS[8]	MAXTRANS[7]	MAXTRANS[6]	MAXTRANS[5]	MAXTRANS[4]	MAXTRANS[3]	MAXTRANS[2]	MAXTRANS[1]	MAXTRANS[0]	MCYCNONREQ[15]	MCYCNONREQ[14]	MCYCNONREQ[13]	MCYCNONREQ[12]	MCYCNONREQ[11]	MCYCNONREQ[10]	MCYCNONREQ[9]	MCYCNONREQ[8]	MCYCNONREQ[7]	MCYCNONREQ[6]	MCYCNONREQ[5]	MCYCNONREQ[4]	MCYCNONREQ[3]	MCYCNONREQ[2]	MCYCNONREQ[1]	MCYCNONREQ[0]
RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW	RpW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-17. MediaLBn AHB Master Control Register (MLBn_AHBMCTL) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	MAXTRANS	Always write "0" to this register. Read value is "X".
[15:0]	MCYCNONREQ	Always write "0" to this register. Read value is "X".

44.2.12 MediaLBn Channel Entry Configuration Register (MLBn_CECR0 - MLBn_CECR15)

MediaLBn Channel Entry Configuration Register (MLBn_CECRn) defines basic attributes for a given logical channel, such as the channel enable, channel type, channel direction, and channel address. There are 16 such registers corresponding to 16 logical MediaLB channels. The definition of the bit fields in the MLBn_CECRn register depends on the selected channel type.

MediaLB Channel Entry Configuration Register (MLBn_CECR0 - MLBn_CECR15)

Figure 44-13. MediaLBn Channel n Entry Configuration Register (MLBn_CECR0)

MLBn_CECR0																															
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**
CE	TR	CT[1]	CT[0]	FCE	MDS[1]	MDS[0]	read0	MASK[7]	MASK[6]	MASK[5]	MASK[4]	MASK[3]	MASK[2]	MASK[1]	MASK[0]	IPL[7]	IPL[6]	IPL[5]	IPL[4]	IPL[3]	IPL[2]	IPL[1]	IPL[0]	CA[7]	CA[6]	CA[5]	CA[4]	CA[3]	CA[2]	CA[1]	CA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-18. MediaLBn Channel n Entry Configuration Register (MLBn_CECR0) bits

Bit Position	Bit Field Name	Bit Description
[31]	CE	Channel Enable Enables channels '0': Channel Disabled '1': Channel enabled
[30]	TR	Channel Transmit Select Sets whether the channel is for transmission or reception. '0': Receive '1': Transmit
[29:28]	CT	Channel Type Select Sets the channel type selection. '00': Synchronous '01': Isochronous, '10': Asynchronous '11': Control

Table 44-18. MediaLBn Channel n Entry Configuration Register (MLBn_CECR0) bits

Bit Position	Bit Field Name	Bit Description
[27]	FCE	Flow Control Enable For Isochronous channel, when set allows an isochronous receive channel to generate the ReceiverBusy (0x10) response. '0': Generation of ReceiverBusy(0x10) response prohibited '1': Generation of ReceiverBusy(0x10) response allowed For Asynchronous and control channel, this bit (PCE) sets whether the reception packet counter should be enabled. This bit is valid for asynchronous and control receive channels in IO Mode only. '0': Reception packet counter disable '1': Reception packet counter enable For Synchronous channel this bit (FSE) sets whether the frame synchronization for the streaming channel should be enabled. '0': Frame synchronization disable '1': Frame synchronization enable
[26:25]	MDS	Channel Mode Select Sets the channel mode '00': DMA mode enable (Ping-pong buffering) '01': DMA mode enable (Circular buffering) ' 10': IO mode enable '11': Reserved. It is prohibited to set MDS[1:0] = '11' Set either DMA mode or IO mode for the channels to be used. All channels must be set to either DMA mode or IO mode. It is prohibited to set the IO mode and DMA mode in a mixed manner.
[24]	read0	-

Table 44-18. MediaLBn Channel n Entry Configuration Register (MLBn_CECR0) bits

Bit Position	Bit Field Name	Bit Description
[23:16]	MASK	Channel interrupt mask MASK[7] - Reserved bit. Read value is '0'. Write always '0' to this bit. MASK[6] - Sets whether channel interrupt due to the frame sync lost should be masked. The status bit targeted for masking is CSCRn:STS[6]. '0': Lost frame synchronization interrupt is not masked '1': Lost frame synchronization interrupt is masked MASK[5] - Reserved bit. Read value is '0'. Write always '0' to this bits. MASK[4] - Sets whether the channel interrupt due to a buffer error should be masked. The status bit targeted for masking is CSCRn:STS[4]. '0': Buffer error channel interrupt is not masked '1': Buffer error channel interrupt is masked MASK[3] - This bit has different interpretation depending on DMA-mode or IO-mode and sets the mask for channel interrupts. The status bit targeted for masking is CSCRn:STS[3]. For DMA mode, it masks the buffer start interrupt. For IO mode, it masks the transmission service request interrupt. '0': Channel interrupt is not masked '1': Channel interrupt is masked MASK[2] - This bit has different interpretation depending on DMA-mode or IO-mode and sets the mask for channel interrupts. The status bit targeted for masking is CSCRn:STS[2]. For DMA mode, it masks the buffer end interrupt. For IO mode, it masks the reception service request interrupt or reception packet abort interrupt. '0': Channel interrupt is not masked '1': Channel interrupt is masked MASK[1] - Sets whether the channel interrupt due to the detection of break should be masked. The status bit targeted for masking is CSCRn:STS[1]. '0': Break detection channel interrupt is not masked '1': Break detection channel interrupt is masked MASK[0] - Sets whether the channel interrupt due a protocol error should be masked. The status bit targeted for masking is CSCRn:STS[0]. '0': Protocol error channel interrupt is not masked '1': Protocol error channel interrupt is masked

Table 44-18. MediaLBn Channel n Entry Configuration Register (MLBn_CECR0) bits

Bit Position	Bit Field Name	Bit Description
[15:8]	IPL	Packet Length For Isochronous channels: For Isochronous transmit channels these bits define the number of packet bytes. The smallest isochronous packet size per frame is 5 bytes (IPL[7:0] >= 5). For Isochronous receive channels, software must program IPL[7:2] to indicate the expected number of bytes per packet, where as IPL[1:0] always equals '00'. A packet length of 8 is indicated by IPL=0x08, a packet length of 12 is indicated by IPL=0x0C, a packet length of 252 is indicated by IPL=0xFC. However, a packet length of 256 bytes is represented as IPL[7:0]=0x00. For Synchronous channel: IPL[7]/FSCD sets frame synchronization disable. '0': Frame synchronization channel is not disable '1': Frame synchronization channel disabled When this bit is set to '1', the channel enable bit (CECRn:CE) is reset to '0' when frame synchronization is lost. IPL[6:5] - These bits are not used. IPL[4:0]/FSPC[4:0] sets the number of frame synchronous physical channels. In other words, it defines the number of physical channels that match with the channel address (CECRn:CA[8:1]) among the synchronous channels contained in one frame. For Asynchronous/Control channel: IPL[7:5] - These bits are not used for Asynchronous/Control channel. IPL[4:0]/PCTH[4:0] sets the packet count threshold. This bit field sets the number of packets to receive before generating a receive packet-count service request. When the number of received packets has reached this setting, a reception service request is generated. Also, when the local channel buffer has become full, a reception service request is generated. When using this bit, it is recommended that the software sets LCBCRn:TH[6:0]=0x00. Set "000" for IPL[7:5]. PCTH[4:0] are valid only in the IO mode.
[7:0]	CA	Channel Address Sets the channel address of the logical channel. These bits determine the ChannelAddress (CA[8:1) associated with this logical channel. This value is matched against the ChannelAddress of each received physical channel from the MediaLB Controller. There is a ChannelAddress match if and only if the ChannelAddress recovered from the MediaLB input, MLBn_MLBSIG, equals the ChannelAddress defined by: CA[15:0] = {7'h00, CA[8:1], 1'b0}

44.2.13 MediaLBn Channel Status Configuration Register (MLBn_CSCR0 - MLBn_CSCR15)

MediaLBn Channel Status Configuration Register (MLBn_CSCRn) reflects the status of the current buffer and previous buffer for the logical channel n. There are 16 such registers corresponding to 16 logical MediaLB channels. The definitions of the bit fields in the MLBn_CSCRn register depends on the selected channel type.

MediaLB Channel Status Configuration Register (MLBn_CSCR0 - MLBn_CSCR15)

Figure 44-14. MediaLBn Channel Status Configuration Register (MLBn_CSCR0)

MLBn_CSCR0																															
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**
BM	BF	read0	read0	read0	read0	read0	read0	read0	read0	read0	read0	IVB[1]	IVB[0]	GB	RDY	STS[3]	STS[2]	STS[1]	STS[0]	STS	STS	STS	STS	STS	STS	STS	STS	STS	STS	STS	STS
Rp	Rp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp	Rp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[31]	BM	Buffer Empty This bit indicates that the local channel buffer is empty. '0': Local channel buffer not empty '1': Local channel buffer empty **Note:** This bit is set and cleared by hardware.
[30]	BF	Buffer Full This bit indicates that the local channel buffer is full. '0': Local channel buffer not full '1': Local channel buffer full **Note:** This bit is set and cleared by hardware.
[29:20]	read0	-

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[19:18]	IVB	Isochronous Valid Bytes These bits are loaded by hardware with the number of valid bytes in the last packet of a broken Isochronous receive channel. Used in conjunction with CCBCRn.BCA, IVB[1:0] can be used by software to determine the final valid byte of the local channel buffer. **Note:** Valid for local channel buffers configured for isochronous RX data. '00': Final valid byte = (CCBCRn.BCA - 5). '01': Final valid byte = (CCBCRn.BCA - 4). '10': Final valid byte = (CCBCRn.BCA - 3). '11': Final valid byte = (CCBCRn.BCA - 2).
[17]	GB	Generate Break This bit has different interpretation depending on the channel configuration. For Synchronous channel, this bit is not used. For Isochronous channel, the function of this bit is to Generate the Isochronous Receive Break (GIRB). When set, this bit causes the hardware to terminate the current packet, flush the local channel buffer, clear the RDY bit, and load CSCRn:IVB[1:0]. This bit is set by system software and cleared by hardware. It is valid for local channel buffers configured for isochronous RX data. For Asynchronous and Control Channel, this bit (GB) enables break generation. When the local channel buffer is configured for transmitting data, the setting of this bit causes the hardware to send the AsyncBreak (0x26) or ControlBreak (0x36) command and stop the transfer. When the local channel buffer is configured for receiving data, the setting of this bit causes hardware to send the MediaLB RxStatus ReceiverBreak (0x70) and stop the transfer. This bit is set by system software and cleared by hardware.
[16]	RDY	Next Buffer Ready This bit has a different interpretation depending on DMA-mode or IO-mode. In IO Mode this bit is not used In DMA Mode, system software should set this bit when all the registers, data and program memory variables are setup and ready to transmit or receive data in DMA Mode. For transmitting data, the system memory buffer should also be filled. For DMA Mode using ping-pong buffering, hardware clears this bit after the buffer begins to be processed. In DMA Mode using circular buffering, the software should clear this bit only when buffer processing needs to halted.

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[15:12]	STS	STS[15-12] Reserved Read value is '0'. Write always '0' to these bits.
[11]	STS	STS[11] This bit has a different interpretation depending on DMA-mode or IO-mode. In DMA mode this is the Previous Buffer Start bit. When set, this bit indicates the first quadlet of the Previous Buffer has been successfully transmitted or received. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all channel types. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software. In IO Mode, this bit is not used.
[10]	STS	STS[10] This bit has a different interpretation depending on DMA-mode or IO-mode. In DMA mode this is the Previous Buffer Done bit. When set, this bit indicates the last quadlet of the Previous Buffer has been successfully transmitted or received. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all channel types. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software. In IO Mode, this bit is not used.

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[9]	STS	STS[9] This bit has different interpretation depending on DMA-mode or IO-mode. In DMA mode this is the Previous Buffer Detect Break bit. When set, this bit indicates that either a transmit channel has detected a receiver break response, ReceiverBreak (0x70), or a receive channel has detected a transmitter break command, ControlBreak (0x36) or AsyncBreak (0x26), while processing the Previous Buffer. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all channel types. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software. In IO Mode this is Receive Packet Start bit. When set, this bit indicates that an RX channel has detected a transmitter packet start command; ControlStart (0x30) or AsyncStart (0x20). This status bit can be used by system software to detect when it has reached the end of an aborted packet. This bit is valid for asynchronous and control RX channels only. **Note:** In IO Mode, the STS[9] bit cannot be programmed to generate a channel interrupt. System software must poll this bit following a Receive Packet Abort (see MLBn_CSCR:STS[8]). As each quadlet of the broken packet is popped from the local channel buffer, software must check to see if the next quadlet is the start of a new packet. When software detects a Receive Packet Start, it can start processing valid data. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[8]	STS	STS[8] This bit has different interpretation depending on DMA-mode or IO-mode. In DMA mode this is Previous Buffer Protocol Error bit. When set, this bit indicates that either a transmit channel has detected an RxStatus of ReceiverProtocolError (0x72), a receive channel has detected an invalid command for this channel type, or an additional AsyncStart (0x20) or ControlStart (0x30) command has been received while in the middle of a packet. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all receive channels and valid for only asynchronous and control transmit channels. In IO mode this is Receive Packet Abort bit. When set, this bit indicates that a receive channel has detected an aborted packet. Received packets are aborted if the receiver generates a break response, ReceiverBreak (0x70), or detects a transmitter packet break command; ControlBreak (0x36) or AsyncBreak (0x26). This bit can also indicate the receive channel has detected a transmit command protocol error. The setting of this bit generates a maskable channel interrupt to system software. This interrupt can be used by system software to detect when it has encountered the beginning of an aborted packet. This bit is valid for asynchronous and control receive channels only. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[7]	STS	STS[7] - Reserved Read value is '0'. Always write '0' to this bits.
[6]	STS	STS[6] - Lost Frame Synchronization bit When set, this bit indicates that the logical channel has lost synchronization with the MediaLB frame. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for synchronous channels only. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[5]	STS	STS[5] This bit has a different interpretation depending on DMA-mode or IO-mode. In DMA mode this is Host Bus Error bit. When set, this bit indicates that an HBI bus error has been detected. The setting of this bit generates a non-maskable channel interrupt to system software. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software. In IO Mode, this bit is not used.

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[4]	STS	STS[4] - Buffer Error bit When set, this bit indicates that either a transmit channel has detected a buffer underflow (e.g., attempted to pop data from an empty buffer), or a receive channel has detected a buffer overflow (e.g., attempted to push data onto a full buffer). The setting of this bit generates a maskable channel interrupt to the system software. This bit is valid for synchronous receive/transmit and isochronous receive (MLBn_CECRn:FCE = 0) channels only. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[3]	STS	STS[3] This bit has different interpretation depending on DMA-mode or IO-mode. In DMA mode this is Current Buffer Start bit. When set, this bit indicates that the DMA controller has started processing the Current Buffer. This bit is set after the contents of MLBn_CNBCRn have been loaded into MLBn_CCBCRn, the MLBn_CSCRn:RDY bit has been cleared (for ping-pong buffering), and hardware is available to accept the next buffer. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all channel types. In IO mode this is a Transmit Service Request bit. When set, it indicates that a transmit channel requests service from the system software. Transmit service requests are issued if the number of valid quadlets in the local channel buffer is less than or equal to MLBn_LCBCRn:TH[6:0]. The setting of this bit generates a maskable channel interrupt to the system software. This bit is valid for all channel types. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.

Table 44-19. MediaLBn Channel Status Configuration Register (MLBn_CSCR0) bits

Bit Position	Bit Field Name	Bit Description
[2]	STS	STS[2] This bit has different interpretation depending on DMA-mode or IO-mode. In DMA mode this is Current Buffer Done bit. When set, this bit indicates that the last quadlet from the last packet (in the Current Buffer) has been successfully transmitted or received. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all channel types. In IO mode this bit is the Receive Service Request bit. When set, this bit indicates that a receive channel is requesting service from system software. Receive service requests are issued if the number of free quadlets in the local channel buffer is less than or equal to MLBn_LCBCRn:TH[6:0]. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for all channel types. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[1]	STS	STS[1] - Current Buffer Detect Break bit. When set, this bit indicates that either a transmit channel has detected a receiver break response, ReceiverBreak (0x70), or a receive channel has detected a transmitter break command, ControlBreak (0x36) or AsyncBreak (0x26), while processing the Current Buffer. The setting of this bit generates a maskable channel interrupt to system software. This bit is valid for asynchronous and control channels only. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.
[0]	STS	STS[0] - Current Buffer Protocol Error bit. This bit indicates that either a transmit channel has detected an RxStatus of ReceiverProtocolError (0x72), a receive channel has detected an invalid command for a given channel type, or an additional ControlStart (0x30) or AsyncStart (0x20) command has been received while in the middle of a packet. The setting of this bit generates a maskable channel interrupt to the system software. This bit is valid for all RX channel types and valid for only asynchronous and control TX channels. Write '1' to clear this bit. Writing '0' has no effect. Once set, this bit holds until it is cleared by software.

44.2.14 MediaLBn Channel Current Buffer Configuration Register (MLBn_CCBCR0 - MLBn_CCBCR15)

MediaLBn Channel n Current Buffer Configuration Register (MLBn_CCBCRn) indicates the address pointer and buffer length of the Current Buffer in system memory for the logical channel n when configured for DMA Mode. When configured for IO Mode it is receive data buffer. There are 16 such registers corresponding to 16 logical MediaLB channels. The definition of the bit fields in the MLBn_CCBCRn register depends on the selected channel type.

MediaLB Channel Current Buffer Configuration Register (MLBn_CCBCR0 - MLBn_CCBCR15)

Figure 44-15. MediaLBn Channel 0 Current Buffer Configuration Register

MLBn_CCBCR0																															
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**
BCA[15]	BCA[14]	BCA[13]	BCA[12]	BCA[11]	BCA[10]	BCA[9]	BCA[8]	BCA[7]	BCA[6]	BCA[5]	BCA[4]	BCA[3]	BCA[2]	BCA[1]	BCA[0]	BFA[15]	BFA[14]	BFA[13]	BFA[12]	BFA[11]	BFA[10]	BFA[9]	BFA[8]	BFA[7]	BFA[6]	BFA[5]	BFA[4]	BFA[3]	BFA[2]	BFA[1]	BFA[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-20. MediaLBn Channel 0 Current Buffer Configuration Register

Bit Position	Bit Field Name	Bit Description
[31:16]	BCA	Buffer Current Address This bit field has different interpretations for DMA mode and IO mode In DMA mode, the BCA field defines a 16-bit address pointer, which identifies the lower half of the beginning address of the Current Buffer in system memory. The BCA[15:2] bits are loaded from MLBn_CNBCRn.BSA[15:2] when the Next Buffer is ready for processing. This Current Buffer address pointer, except when associated with isochronous channels, should always be quadlet aligned (i.e., set BCA[1:0] to '00' for Synchronous, Asynchronous and Control channels). During the processing of the Current Buffer, the BCA field marks which quadlet of the buffer is currently being processed. The upper half of the beginning address of the Current Buffer is system memory is defined by MLBn_SBCR.SRBA, MLBn_ABCR.ARBA, MLBn_CBCR.CRBA, or MLBn_IBCR.IRBA when MLBn_CECRn.TR is clear; MLBn_SBCR.STBA, MLBn_ABCR.ATBA, MLBn_CBCR.CTBA, or MLBn_IBCR.ITBA when MLBn_CECRn.TR is set, dependant on the value of MLBn_CECRn.CT[1:0]. In IO mode, this bit field defines the Receive Data Buffer bits - RDB[31:16]. This field contains the upper half of the next quadlet of receive data when the logical channel is configured as receive channel.

Table 44-20. MediaLBn Channel 0 Current Buffer Configuration Register

Bit Position	Bit Field Name	Bit Description
[15:0]	BFA	Buffer Final Address This bit field has different interpretations for DMA mode and IO mode In DMA mode, the BFA field defines a 16-bit address pointer, which identifies the lower half of the ending address of the Current Buffer in system memory. The BFA[15:2] bits are loaded from MLBn_CNBCRn.BEA[15:2] when the Next Buffer is read for processing. This Current Buffer address pointer, except when associated with isochronous channels, should always be quadlet aligned (i.e., BFA[1:0] equals '00' for Synchronous, Asynchronous and Control channels). During the processing of the Current Buffer, the point at which the BCA field becomes equal to (or greater than) the BFA field indicates that the processing of the Current Buffer will end upon successful completion of the current quadlet (for isochronous and synchronous channels) or upon successful completion of the current packet (for asynchronous and control channels). It is the responsibility of system software to ensure the system memory buffers (for RX asynchronous and control channels) can accommodate overflow in the size of the largest packet supported. Additionally, single-packet buffering can be used by simply programming MLBn_CNBCRn.BSA[15:2] = MLBn_CNBCRn.BEA[15:2]. The upper half of the ending address of the Current Buffer in system memory is defined by MLBn_SBCR.SRBA, MLBn_ABCR.ARBA, MLBn_CBCR.CRBA, and MLBn_IBCR.IRBA when MLBn_CECRn.TR is clear; MLBn_SBCR.STBA, MLBn_ABCR.ATBA, MLBn_CBCR.CTBA, or MLBn_IBCR.ITBA when MLBn_CECRn.TR is set, dependant on the value of MLBn_CECRn.CT[1:0]. In IO mode, this bit field defines the Receive Data Buffer bits - RDB[15:0]. This field contains the lower half of the next quadlet of receive data when the logical channel is configured as receive channel.

44.2.15 MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0 - MLBn_CNBCR15)

The Channel n Next Buffer Configuration Register (MLBn_CNBCRn) configures the start and end addresses of the Next Buffer in system memory for the logical channel when configured for DMA Mode. When configured for IO Mode, the MLBn_CNBCRn registers is the Transmit Data Buffer. There are 16 such registers corresponding to 16 logical MediaLB channels. The definition of the bit fields in the MLBn_CNBCRn register depend on the selected channel type.

MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0 - MLBn_CNBCR15)

Figure 44-16. MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0)

MLBn_CNBCR0																															
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**
BSA[15]	BSA[14]	BSA[13]	BSA[12]	BSA[11]	BSA[10]	BSA[9]	BSA[8]	BSA[7]	BSA[6]	BSA[5]	BSA[4]	BSA[3]	BSA[2]	BSA[1]	BSA[0]	BEA[15]	BEA[14]	BEA[13]	BEA[12]	BEA[11]	BEA[10]	BEA[9]	BEA[8]	BEA[7]	BEA[6]	BEA[5]	BEA[4]	BEA[3]	BEA[2]	BEA[1]	BEA[0]
RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-21. MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0) bits

Bit Position	Bit Field Name	Bit Description
[31:16]	BSA	Buffer Start Address bits This bit field has different interpretations for DMA mode and IO mode. In DMA mode, the BSA field defines a 16-bit address pointer, which identifies the lower half of the beginning address of the Next Buffer in system memory. Once system software detects MLBn_CSCRn.RDY has been cleared by hardware (for ping-pong buffering), the beginning address of the Next Buffer may be loaded into BSA[15:2]. System software should then set MLBn_CSCRn.RDY. Once processing of the Current Buffer for the logical channel is complete, the BSA[15:2] field is loaded into the MLBn_CCBCRn.BCA[15:2] field and processing of the next buffer can begin. This Next Buffer address pointer must always be quadlet aligned (e.g., BSA[1:0] must be written as '00'). The upper half of the beginning address of the Next Buffer in system memory is defined by MLBn_SBCR.SRBA, MLBn_ABCR.ARBA, MLBn_CBCR.CRBA, or MLBn_IBCR.IRBA when MLBn_CECRn.TR is clear; MLBn_SBCR.STBA, MLBn_ABCR.ATBA, MLBn_CBCR.CTBA, or MLBn_IBCR.ITBA when MLBn_CECRn.TR is set, dependant on the value of MLBn_CECRn.CT[1:0]. Note - For BSA[1:0] bits in DMA mode, read value is 'X'. Write always '0' to these bits. In IO mode, this bit field defines the Transmit Data Buffer bits - TDB[31:16]. This field contains the upper half of the next quadlet of transmit data when the logical channel is configured as transmit channel.

Table 44-21. MediaLBn Channel Next Buffer Configuration Register (MLBn_CNBCR0) bits

Bit Position	Bit Field Name	Bit Description
[15:0]	BEA	Buffer End Address bits This bit field has different interpretations for DMA mode and IO mode In DMA mode, the BEA field defines a 16-bit address pointer, which identifies the lower half of the ending address of the Next Buffer in system memory. Once system software detects MLBn_CSCRn.RDY has been cleared by hardware (for ping-pong buffering), the ending address of the Next Buffer may be loaded into BEA[15:2]. System software should then set MLBn_CSCRn.RDY. Once processing of the Current Buffer for the logical channel is complete, the BEA[15:2] field is loaded into the MLBn_CCBCRn.BFA[15:2] field and processing of the next buffer can begin. The BEA[15:2] bits are loaded into MLBn_CCBCRn.BFA[15:2] when the Current Buffer is finished being processed. This Next Buffer address pointer, except when associated with isochronous channels, should always be quadlet aligned (i.e., set BEA[1:0] to '00' for Synchronous, Asynchronous and Control channels). The upper half of the ending address of the Next Buffer in system memory is defined by MLBn_SBCR.SRBA, MLBn_ABCR.ARBA, MLBn_CBCR.CRBA, and MLBn_IBCR.IRBA when MLBn_CECRn.TR is clear; MLBn_SBCR.STBA, MLBn_ABCR.ATBA, MLBn_CBCR.CTBA, or MLBn_IBCR.ITBA when MLBn_CECRn.TR is set, dependant on the value of MLBn_CECRn.CT[1:0]. In IO mode, this bit field defines the Transmit Data Buffer bits - TDB[15:0]. This field contains the lower half of the next quadlet of transmit data when the logical channel is configured as transmit channel.

44.2.16 MediaLBn Local Channel Buffer Configuration Register (MLBn_LCBCR0 - MLBn_LCBCR15)

MediaLBn Local Channel n Buffer Configuration Register (MLBn_LCBCRn) configures the allocation of the local channel buffer area on the RAM for the local channel buffer. There are 16 such registers corresponding to 16 logical MediaLB channels. This register should only be written while the logical channel is disabled. Writing to this register while the corresponding logical channel is enabled may result in unexpected behavior.

MediaLB Local Channel Buffer Configuration Register (MLBn_LCBCR0 - MLBn_LCBCR15)

Figure 44-17. MediaLBn Local Channel 0 Buffer Configuration Register (MLBn_LCBCR0)

MLBn_LCBCR0																															
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**
read0	read0	read0	TH[6]	TH[5]	TH[4]	TH[3]	TH[2]	TH[1]	TH[0]	read0	read0	read0	BD[5]	BD[4]	BD[3]	BD[2]	BD[1]	BD[0]	read0	read0	read0	read0	read0	read0	read0	SA[5]	SA[4]	SA[3]	SA[2]	SA[1]	SA[0]
Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	Rp0	RpWp	RpWp	RpWp	RpWp	RpWp	RpWp
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-22. MediaLBn Local Channel 0 Buffer Configuration Register (MLBn_LCBCR0) bits

Bit Position	Bit Field Name	Bit Description
[31:29]	read0	-
[28:22]	TH	Buffer Threshold TH[6:0] sets the threshold of the local channel buffer. The buffer threshold is set in units of 2 quadlets. This bit is used to determine whether a transmission/reception service request should be issued. '0x00': When TH[6:0] is 0x00, an reception service request is generated if the buffer is full and a transmission service request is generated when the buffer is empty. '0x01': Threshold = 2 quadlets '0x02': Threshold = 4 quadlets '0x03': ..... '0x7F': Threshold = 254 quadlets
[21:19]	read0	-

Table 44-22. MediaLBn Local Channel 0 Buffer Configuration Register (MLBn_LCBCR0) bits

Bit Position	Bit Field Name	Bit Description
[18:13]	BD	Buffer Depth BD[5:0] sets the depth of the local channel buffer. The depth is set in units of 4 quadlets. '0x00': Depth= 4 quadlets '0x01': Depth= 8 quadlets '0x02': Depth= 12 quadlets '0x03': ..... '0x3F': Depth= 256 quadlets
[12:6]	read0	-
[5:0]	SA	Buffer Start Address SA[5:0] sets the start address of the local channel buffer. '0x00': RAM Start Address offset of 0 quadlets '0x01': RAM Start Address offset of 4 quadlets '0x02': RAM Start Address offset of 8 quadlets '0x03': ..... '0x3F': RAM Start Address offset of 252 quadlets **Note:** The initial value of SA[5:0] is n*4, where n is the channel number (0,1,2 ... 15). The initial value of SA[5:0] for Channel 0 is '0x0', for Channel 1 is '0x4', for Channel 2 is '0x8' and so on.

44.2.17 MediaLBn Module Identification Register (MLBn_MID)

This register identifies the particular version of the MediaLB hardware used in the device.

MediaLB Module Identification Register (MLBn_MID)

Figure 44-18. MediaLBn Module Identification Register (MLBn_MID)

MLBn_MID																															
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**
MID[31]	MID[30]	MID[29]	MID[28]	MID[27]	MID[26]	MID[25]	MID[24]	MID[23]	MID[22]	MID[21]	MID[20]	MID[19]	MID[18]	MID[17]	MID[16]	MID[15]	MID[14]	MID[13]	MID[12]	MID[11]	MID[10]	MID[9]	MID[8]	MID[7]	MID[6]	MID[5]	MID[4]	MID[3]	MID[2]	MID[1]	MID[0]
Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp	Rp
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44-23. MediaLBn Module Identification Register (MLBn_MID) bits

Bit Position	Bit Field Name	Bit Description
[31:0]	MID	Module ID The MediaLB module implemented in the device may vary from device to device. This register identifies the particular version of the hardware used in the device. This register helps in developing software to the hardware version implemented in the device. **Note:** Please refer to the device specific data sheet for the MID value

44.3 Notes on Using MediaLB

This section is the “Programmer’s Guide”, which lists the usage notes for programming the MediaLB module. It is recommended to contact SMSC for further details.

General Usage Notes

- Reserved bits return undefined values. The software programs shall be independent of the values read from the reserved register bits.
- If the MediaLB should be accessible in user mode, PPU should be configured accordingly.

Steps in programming the MediaLB Module

- After the system reset, the software shall detect the Module ID number of MediaLB, by reading the MLBn_MID register. This would help it in identifying the attributes and capabilities supported by the MediaLB module.
- Then Software should configure MediaLB by setting appropriate registers following recommended Software Flowcharts. For further details, please contact SMSC.

44.4 List of References

This chapter lists the various references to other documents, used in this specification.

- OS62400 MediaLB Device Interface Macro Advanced Product Data Sheet (DS62400AP5) by SMSC, issued on October 2006.
- MediaLB Specification by SMSC (TB0400AN3V0, rev 3.0 issued on February 2006)

Revision History

Revision History

Document Title: FCR4 Cluster Series Hardware Manual Document Number: 002-09388				
Revision	**ECN#**	**Issue Date**	**Origin of Change**	**Description of Change**
**	-	-	NOFL	Initial release as Spansion MN707-00001-1v2-E.
*A	5885258	10/17/2017	SSAS	Migrated to Cypress format.
*B	6070862	02/14/2018	DENA	Updated notes of Table 16-26 and Table 16-28. □ MB9DF126 (“Calypso”) --> MB9DF126 (“Atlas”) □ MB9EF126 (“Atlas”) --> MB9EF126 (“Calypso”) Updated 1. and 3. in 43.5 References 1: SHE Functional Specification v1.1 (rev 439) can be requested at from OEM HIS group 3: SP 800-38 A -- Recommendation for Block Cipher Modes of Operation: Methods and Techniques: https://csrc.nist.gov/publications/detail/sp/800-38a/final
*C	6532968	04/04/2019	NOFL	Deleted MB9EF126 Calypso Renamed product name from prefix MB to prefix CY.